MASTER EN SCIENCES INFORMATIQUES

Médecine nucléaire

Amélioration d'un outil de traitement pour la segmentation d'images médicales 2D et 3D dans le domaine de l'ostéoarticulaire

Michiels, Y

Award date:
2012

Awarding institution:
Universite de Namur

FUNDP
INSTITUT D'INFORMATIQUE
Rue Grandgagnage, 21
5000 NAMUR (BELGIUM)

CLINIQUES UNIVERSITAIRES
MONT-GODINNE
Avenue du Docteur Gaston Thérasse, 1
5530 YVOIR (BELGIUM)

MEDECINE NUCLEAIRE

Médecine Nucléaire : Amélioration d'un outil de traitement pour la segmentation d'images médicales 2D et 3D dans le domaine de l'ostéoarticulaire

Michiels Yvan

Promoteur : Jean-Paul LECLERCQ
Co-promoteur : Hubert MEURISSE

Année académique 2011-2012
Mémoire présenté en vue de l'obtention du grade de Master en Sciences Informatiques

Résumé

La segmentation d'images tridimensionnelles à caractère médical doit permettre de visualiser les différentes structures anatomiques internes, de les mesurer, de détecter et localiser sur celles-ci les anomalies ainsi que de les quantifier. Dans le cadre particulier de l'ostéoarticulaire, la segmentation doit permettre de diagnostiquer rapidement les lésions cartilagineuses par comparaison entre deux examens de type arthro-scanner. À l'heure actuelle, le processus de segmentation est principalement réalisé manuellement par un expert. C'est une opération couteuse en temps et dont le résultat dépend de l'environnement (fatigue, réglages de l'écran, éclairage,…) et de l'expert (concentration, aisance avec l'outil, patience,…). L'objectif principal de ce travail est d'améliorer un logiciel de traitement d'images en cours de développement, de le rendre plus ergonomique et plus efficace dans la segmentation. Cet outil permettra de déterminer le volume du cartilage en demandant un minimum d'interventions manuelles afin d'être aussi indépendante possible de l'utilisateur.

Mots-clés : *arthro-scanner, image tridimentionnelle, segmentation, ostéoarticulaire, cartilage, volume*

Abstract

The segmentation of tridimensional medical images should enable to visualize the various internal anatomical structures, to measures these, to detect, localize and quantify anomalies on these. In the particular context of osteoarticular, the segmentation should enable quickly diagnose of cartilaginous injuries driven by an analysis between two arthroscan examinations. At present, the segmentation process is primarily done manually by an expert. This is a time expensive operation and the result depends on the environment (fatigue, screen settings, lighting,...) and the expert (concentration, ease with the tool, patience ...). The main objective of this work is to improve an image processing software under development, making it more ergonomic and efficient in segmentation. This tool will determine the cartilage volume by requiring a minimum of manual intervention to be as independent as possible from the user.

Keywords : *arthroscan, cartilage, segmentation, tridimensional image, osteoarticular, volume*

Remerciements

Comme préambule à ce mémoire, j'aimerais remercier l'ensemble des personnes qui m'ont apporté leur aide pendant le déroulement du stage ainsi que la rédaction de ce mémoire.

Dans un premier temps j'aimerais remercier tout naturellement Mr Jean-Paul Leclercq, professeur aux Facultés Universitaires Notre Dame de la Paix de Namur et promoteur de ce mémoire. Sa disponibilité, la rapidité de ses réponses et ses conseils judicieux m'ont été d'une très grande aide.

Je remercie aussi Mr Hubert Meurisse, maître de stage et co-promoteur de ce travail. Son accueil chaleureux m'a permis de me sentir à l'aise. Bien que très demandé, il a toujours été disponible pour répondre à mes questions et m'a laissé une très grande liberté dans les choix effectués.

Je tiens également à remercier l'ensemble du personnel du service de médecine nucléaire des Cliniques universitaires de Mont-Godinne. Ils ont toujours été accueillants et très chaleureux. L'ambiance de travail a toujours été des plus conviviales.

Je remercie aussi les développeurs de Mont-Godinne qui m'ont apporté leur aide et de nouvelles voies de recherche lorsque je n'en voyais plus.

Comme débutant dans le développement de logiciels en imagerie médicale, j'aimerais remercier la communauté ITK et VTK, pour leur aide dans la compréhension et l'utilisation des librairies ITK et VTK, ainsi que l'équipe du site du zéro, offrant des tutoriaux de qualité pour la prise en main de la librairie Qt.

Au terme de mes études, je tiens également à remercier l'ensemble des assistants et professeurs de la faculté d'informatique de Namur. Ils offrent un enseignement de qualité et sont toujours accessibles et sympathiques.

De manière plus personnelle, j'aimerais remercier mes parents qui m'ont soutenu tout au long de mes études et qui m'ont permis de faire de telles études. Enfin, je remercie ma chère et tendre Marie-Laure qui m'a supporté tout au long de ce travail et qui m'a apporté sa connaissance de la langue française pour la relecture de ce mémoire.

Merci à toutes et à tous.

Table des matières

Table des figures

Introduction

Au fil des ans, la médecine n'a cessé de se développer et de se perfectionner, tant au niveau des interventions qu'au niveau du diagnostic. L'imagerie médicale est un domaine important de la médecine actuelle qui a rendu possible l'examen interne du corps humain sans avoir à l'inciser. Il existe plusieurs méthodes d'imagerie médicale, chacune ayant son domaine de prédilection. Parmi les plus connues, nous pouvons citer l'imagerie par résonance magnétique, la tomographie par émission de positrons, l'imagerie par rayon X,… De nos jours, l'imagerie médicale est présente aussi bien dans le diagnostic, le suivi thérapeutique et dans la planification du traitement.

À l'heure actuelle, le traitement automatique des images médicales est encore loin d'être entièrement atteint, de nombreuses tâches étant encore exécutées manuellement par le médecin. Ainsi lorsque le médecin observe une image, il procède mentalement à l'identification des différents organes et par la même occasion à la division de l'image en régions représentant ces organes. L'automatisation ou semi-automatisation de ce processus d'identification des tissus et organes par une machine permettrait de gagner un temps non négligeable et permettrait de réduire le risque d'erreur commise par le médecin. L'automatisation de ces processus appartient bien sûr au traitement d'image mais plus particulièrement au domaine de la segmentation d'image. Dans le cadre de ce mémoire, la segmentation d'image est une analyse de l'image permettant d'identifier certains organes.

La segmentation manuelle, bien que encore très utilisée, nécessite une quantité considérable de temps, notamment par la précision du travail demandé ainsi que par la quantité d'images à traiter. En plus de requérir énormément de temps, cette technique a le désavantage de présenter des fluctuations parfois importantes dans le résultat, aussi bien pour des experts différents que pour le même expert. En effet, les conditions de travail (par exemple l'éclairage) et l'état de fatigue de l'expert peuvent avoir un effet sur le résultat. Le faible contraste existant parfois entre différentes parties de l'image rend difficile l'identification de la frontière entre les organes. Ce dernier fait est aussi une raison de la fluctuation des résultats. La segmentation manuelle est donc difficilement reproductible.

Pour répondre à ce problème majeur (la reproductibilité de la segmentation), un grand nombre de méthodes de segmentation automatique et semi automatique ont vu le jour en peu de temps. Cette apparition massive de méthodes de segmentation s'explique par le fait que chacune des méthodes est spécifique à un organe sur un ou deux type(s) d'images médicales. Vu la quantité astronomique d'informations présentes sur une image médicale et les variations importantes entre les patients, la mise au point d'un outil de segmentation universel d'image médicale représente un travail titanesque. C'est pourquoi les outils de segmentation développés pour le moment se limitent à un organe pour un type d'acquisition. Nous en ferons de même dans ce mémoire. Nous nous limiterons aux images de genou acquises par un arthro-scanner et à la segmentation du cartilage de cette même articulation. Dans ce cadrage du travail, nous nous intéresserons plus particulièrement aux coupes dites sagittales et au cartilage fémoral. Cette segmentation présente un intérêt certain dans le diagnostic de l'arthrose du genou.

Dans le premier chapitre de ce mémoire, nous présenterons le cadre de notre travail, c'est-à-dire l'articulation du genou, le cartilage ainsi que l'arthrose du genou. Nous parlerons aussi de l'imagerie médicale et plus particulièrement de la méthode d'acquisition utilisée dans ce domaine.

Nous parlerons, au chapitre suivant, des objectifs fixés pour le travail qui en seront le fil conducteur.

Dans le chapitre intitulé *Matériel et méthode,* le lecteur pourra prendre connaissance de tous les moyens (informatiques, techniques) utilisés pour accomplir les objectifs.

Les solutions apportées aux objectifs pendant le stage dans le service de médecine nucléaire aux Cliniques universitaires de Mont-Godinne, seront alors présentées dans le chapitre *Application et résultats.*

Le cinquième chapitre du mémoire est réservé au programmeur souhaitant continuer le développement de l'outil fourni, il présente la conception de l'outil afin de mieux le prendre en main.

Enfin, le dernier chapitre donne diverses perspectives de développement de l'outil, aussi bien au niveau de l'ergonomie que sur les performances de temps et de résultats.

Chapitre 1 – Cadre

Ce chapitre a pour objectif de fixer le cadre de ce mémoire : l'imagerie médicale et le cartilage du genou. L'imagerie médicale est un domaine complexe qui utilise de nombreux standards. Ce chapitre n'a pas la prétention d'expliquer l'intégralité de l'imagerie médicale. Nous nous contenterons de présenter les concepts nécessaires pour la compréhension du domaine de recherche.

Concernant le genou, nous en énoncerons l'anatomie de façon ciblée pour notre problème et les pathologies liées au cartilage.

1.1 Le genou

Le genou est une articulation complexe, il est à la fois souple et solide. Il s'agit de l'articulation qui relie la jambe à la cuisse et qui permet l'extension et la flexion de la jambe. Il supporte le poids du corps et permet de pratiquer de multiples activités sportives.

1.1.1 Anatomie

L'articulation du genou met en jeu trois os : le Fémur, le Tibia et la Rotule. Le cartilage présent sur ces trois os assure la fluidité de mouvement de l'articulation. Le cartilage forme une couche protectrice sur les surfaces des os formant l'articulation, leur permettant de glisser facilement les unes sur les autres. Les informations fournies dans cette section sont principalement tirées de l'ouvrage de K. Moore et A. Dalley [1].

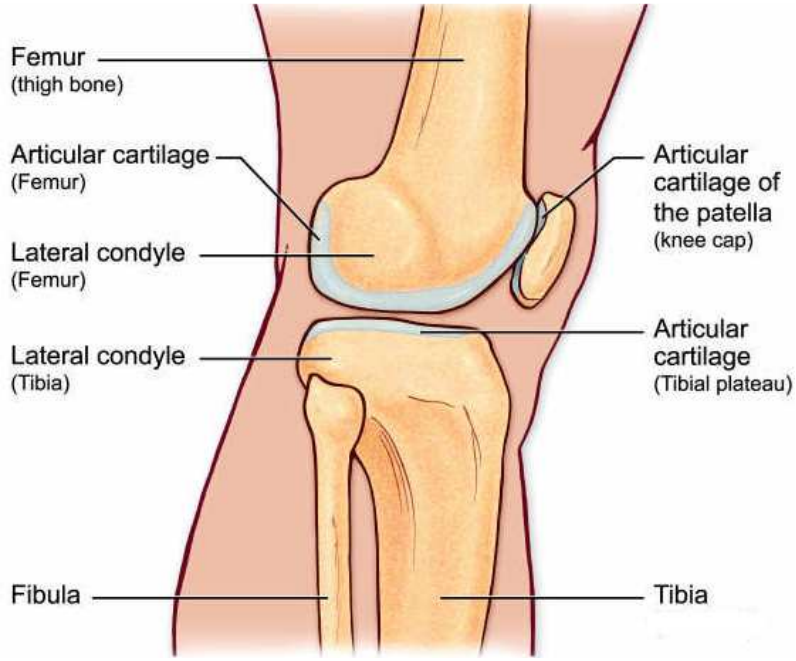


Figure 1 - Anatomie du genou (os et cartilage)

Le genou est aussi composé de ligaments (présents sur la figure suivante). Il y en a quatre, le ligament latéral externe, le ligament latéral interne et les ligaments croisés (LCP et LCA[1])

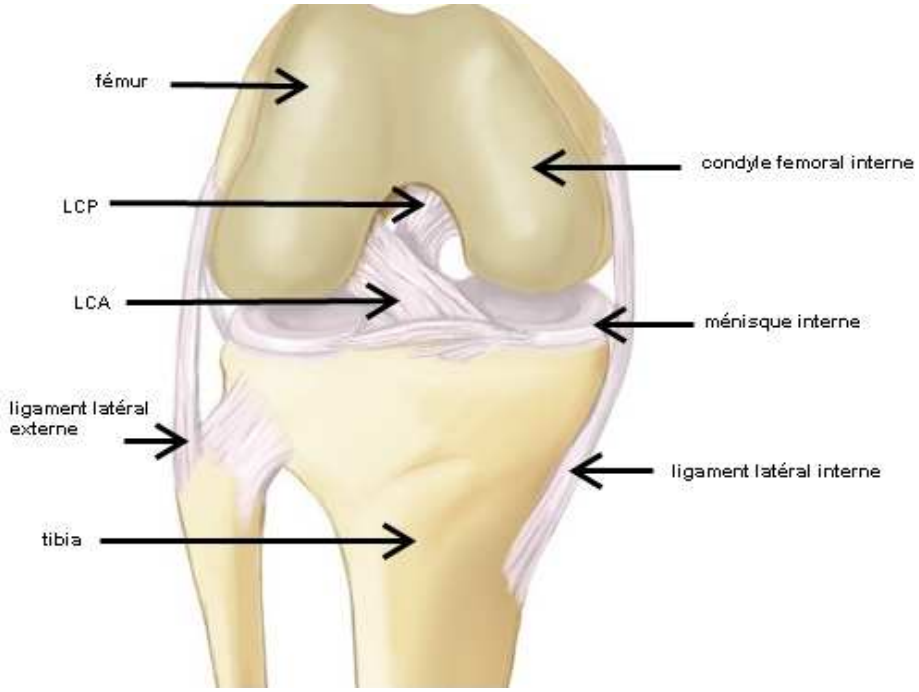


Figure 2 - Anatomie du genou avec ligaments

L'extrémité inférieure du fémur présente deux surfaces articulaires en forme de poulies, les condyles fémoraux externe et interne. L'extrémité supérieure du tibia présente deux surfaces articulaires concaves sur lesquelles les condyles fémoraux viennent se poser, le plateau tibial. Ces surfaces articulaires sont recouvertes de cartilage. Entre les deux, s'interposent deux ménisques interne et externe, sorte de coussinets de cartilage, jouant le rôle d'amortisseurs. En avant se situe la rotule ou patella, de forme triangulaire. On définit ainsi trois compartiments anatomiques :

- le compartiment fémoro-tibial interne ;
- le compartiment fémoro-tibial externe ;
- le compartiment fémoro-patellaire.

1.1.2 Pathologies

L'articulation du genou peut souffrir de nombreuses pathologies. Toutefois, nous nous intéresserons principalement aux pathologies liées au cartilage. En particulier nous pouvons citer la gonarthrose ou arthrose du genou. C'est la cause de douleur dans le genou la plus fréquente chez les personnes de plus de 50 ans. La gonarthrose est par ailleurs la forme d'arthrose la plus fréquente.

Comme nous nous intéressons au cartilage, nous allons préciser sa nature. Le cartilage est un tissu spécialisé composé de cellules de forme arrondie appelées chondrocytes. Il existe trois types différents de cartilage dont les compositions changent de par leurs fonctions [2] :

- Le cartilage élastique : ce cartilage est jaunâtre et maintient la forme d'une structure en lui offrant une grande flexibilité. Il est présent dans l'oreille ou le larynx.

[1] LCP pour ligament croisé postérieur et LCA pour ligament croisé antérieur.

- Le cartilage hyalin : les cellules qui y sont présentes sont volumineuses. Ce cartilage se retrouve en surface des articulations de type synoviale, telle que le genou.
- Le cartilage fibreux ou fibrocartilage : ce cartilage est riche en fibre et est très résistant aux tractions. Il compose notamment les ménisques du genou

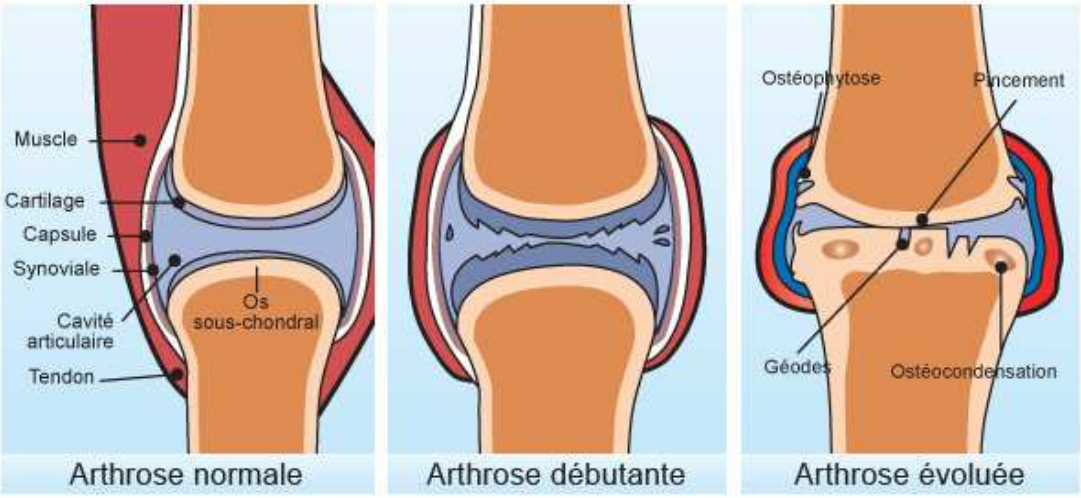


Figure 3 - Évolution de l'arthrose du genou

Nous retrouvons dans le genou deux types de cartilages : le cartilage hyalin, qui recouvre le fémur, la rotule et une partie du tibia, et le cartilage fibreux, présent sur le tibia pour former deux coussinets appelés ménisques.

Le cartilage s'use au fil des ans en fonction de son utilisation et s'usera davantage s'il est fort sollicité. Même si le cartilage peut se reconstituer, il ne possède pas de grande capacité de régénération, trois mois lui sont nécessaire pour se renouveler contrairement à d'autres organes qui ne prennent que quelques jours. Ceci s'explique par l'absence de vaisseaux sanguins. De plus lorsque le cartilage guéri, le tissu cicatriciel se compose essentiellement de cartilage fibreux, de moins bonne qualité que le cartilage hyalin.

La gonarthrose se définit comme un amincissement de la couche de cartilage. En fonction de l'emplacement touché par l'arthrose, nous parlerons d'arthrose fémoro-tibial latéral, d'arthrose fémoro-tibiale médiale, d'arthrose fémoro-patellaire ou global si toutes les parties sont touchées.

L'apparition d'arthrose dans le genou peut avoir diverses causes. Il peut apparaître suite à un traumatisme direct (tel qu'une fracture du fémur), indirect (comme par exemple une entorse du genou) ou suite à des microtraumatismes dus à une activité répétée (sport ou métier). Des défauts génétiques de fabrication du cartilage peuvent également en être la cause [3].

Figure 4 - Genou abimé

Cette affection peut aussi être causée par un mauvais alignement du fémur et du tibia. Le fémur est placé en diagonale au sein de la cuisse, tandis que le tibia est presque vertical au sein de la jambe ; les axes longitudinaux des deux os délimitent donc un angle au niveau du genou. Cet angle est appelé angle Q. Lorsque l'angle est normal, l'angulation du fémur au sein de la cuisse place le milieu de l'articulation du genou juste en dessous de la tête du fémur en position debout.

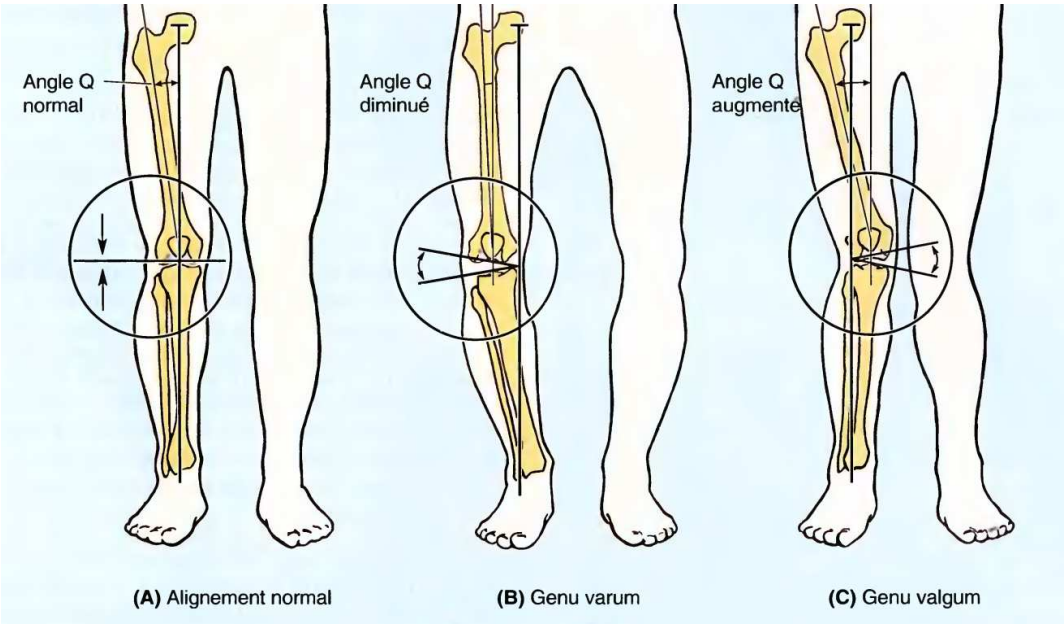


Figure 5 - Genu varum et Genu Valgum

Le *genu varum* est une difformité dans laquelle l'angle Q est plus petit car la direction du fémur dans la cuisse est anormalement verticale ; il existe dans ce cas une angulation médiale de la jambe par rapport à la cuisse (jambes arquées) responsable d'un déséquilibre dans la transmission du poids du corps.

Une angulation latérale excessive de la cuisse par rapport à la jambe est appelée *genu valgum* (genoux cagneux) ; le centre de gravité en position debout tombe dans ce cas latéralement

par rapport au centre du genou. En conséquence, c'est le ligament collatéral tibial qui est soumis à des contraintes excessives, de même que le ménisque latéral et le cartilage articulaire des condyles latéraux du fémur et du tibia.

1.1.3 Traitements

Différents traitements sont possibles en fonction de la cause de l'arthrose et de l'état d'avancement de la dégradation du cartilage.

Dans les premiers stades de l'arthrose, un traitement médicamenteux suffit. Lors de lésions, la prise d'analgésiques et d'anti-arthrosiques est requise. Des lavages articulaires et des infiltrations peuvent être utiles.

En ce qui concerne l'arthrose de la partie externe du genou et de la rotule, un traitement médicamenteux associé à une rééducation est indispensable. Lorsque l'arthrose est due à un genu varum, ce qui concerne la partie interne du genou, une intervention chirurgicale est nécessaire afin de rétablir un axe acceptable à la jambe.

Dans les cas les plus graves, une intervention visant à mettre en place une prothèse est envisageable, surtout lorsque la marche est impossible.

1.2 L'imagerie médicale

L'imagerie médicale regroupe les différents moyens d'acquisitions et de restitutions d'images du corps humain à partir de phénomènes physiques tels que l'absorption de rayons X, la résonnance magnétique, la réflexion d'ultra-sons ou la mesure de la radioactivité. D'autres méthodes d'imagerie optique, comme l'endoscopie, sont parfois associées à ces techniques. Les techniques d'imagerie médicale sont donc des procédés permettant d'examiner l'intérieur du corps d'un patient sans devoir l'opérer. [4] [5]

Le but de l'imagerie médicale est de créer une représentation visuelle intelligible d'une information à caractère médical. L'objectif est en effet de pouvoir représenter sous un format relativement simple une grande quantité d'informations issues d'une multitude de mesures acquises selon un mode bien défini. L'image obtenue peut être traitée informatiquement pour obtenir par exemple :

- une reconstruction tridimensionnelle d'un organe ou d'un tissu ;
- un film ou une animation montrant l'évolution ou les mouvements d'un organe au cours du temps ;
- une imagerie quantitative qui représente les valeurs mesurées pour certains paramètres biologiques dans un volume donné.

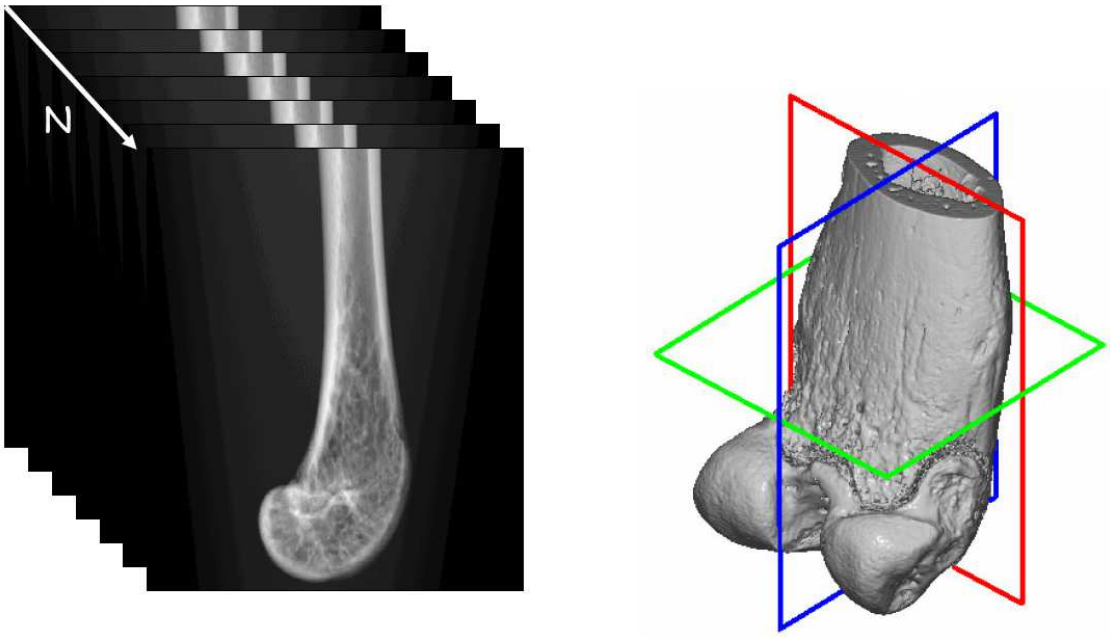


Figure 6 - À gauche, une série d'image 2D; À droite, le volume 3D correspondant

Dans un sens plus large, le domaine de l'imagerie médicale englobe toutes les techniques permettant de stocker et de manipuler les informations fournies par de telles acquisitions. Ainsi, il existe une norme pour la gestion informatique des données issues de l'imagerie médicale : la norme DICOM.

1.2.1 Les techniques

Nous l'avons déjà mentionné, l'imagerie médicale regroupe toute une série de techniques. Suivant les techniques utilisées, les examens permettent d'obtenir des informations sur l'anatomie des organes (leur taille, leur volume, leur localisation, la forme d'une éventuelle lésion, etc.). On parlera alors d'imagerie structurelle. D'autres techniques fournissent quant à elles des informations sur le fonctionnement des organes (leur physiologie, leur métabolisme, etc.), on parle dans ce cas d'imagerie fonctionnelle.

Parmi les méthodes d'imagerie structurelle employées en médecine, nous pouvons citer d'une part les méthodes basées soit sur les rayons X (radiologie conventionnelle, radiologie digitale, tomodensitomètre ou CT-scan, angiographie, etc.) soit sur la résonance magnétique nucléaire (IRM), les méthodes échographiques (qui utilisent les ultra-sons), et enfin les méthodes optiques (qui utilisent les rayons lumineux).

Les méthodes d'imagerie fonctionnelle sont aussi très variées. Elles regroupent les techniques de médecine nucléaire (TEP, TEMP) basées sur l'émission de positons ou de rayons gamma par des traceurs radioactifs qui, après injection, se concentrent dans les régions d'intense activité métabolique, les techniques électro-physiologiques qui mesurent les modifications de l'état électrochimique des tissus (en particulier en lien avec l'activité nerveuse), les techniques issues de l'IRM dite fonctionnelle ou encore les mesures thermographiques ou de spectroscopie infrarouge.

La multiplication des techniques et leur complémentarité poussent les progrès dans la direction d'une imagerie dite multimodale dans laquelle les données issues de plusieurs technologies

sont mises en communs et recalées, c'est-à-dire mises en correspondance au sein d'un même document. Cela permet un apport supplémentaire d'informations au sein d'un même fichier.

1.2.2 La visualisation

Les techniques citées ci-dessus permettent d'acquérir les images par coupes. Ces coupes représentent une certaine épaisseur du corps observé. L'ensemble des coupes permettent de donner une idée générale du volume du corps observé.

Pour aller plus loin dans la visualisation, par un traitement informatique, il est possible de reconstruire un volume à partir de l'ensemble des coupes. Ainsi, il est possible d'observer un organe dans son ensemble sans devoir passer d'une coupe à l'autre.

Toutefois, la visualisation par coupes permet de voir plus de détails. En effet, dans la visualisation d'un volume, l'intérieur de l'organe est difficilement observable. De plus, il est possible de visualiser des coupes selon des axes différents. Nous allons énoncer les trois axes utilisés en médecine pour la visualisation d'image.

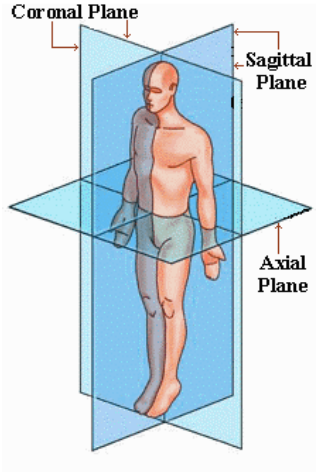


Figure 7 - Système de plan anatomique

- Le plan coronal : ce plan sépare le corps en une partie antérieure (côté torse) et une partie postérieure (côté dos) ;
- Le plan axial : ce plan divise le corps en une partie supérieure (côté tête) et une partie inférieure (côté pieds) ;
- Le plan sagittal : ce dernier plan sépare le côté gauche du corps du côté droit.

1.3 CT-Scan

La tomodensitométrie (TDM), dite aussi scanographie, CT-scan[2], ou simplement scanner pour l'appareil, est une technique d'imagerie médicale qui consiste à mesurer l'absorption des rayons X par les tissus puis, par traitement informatique, à numériser et enfin reconstruire des images 2D ou 3D des structures anatomiques. Pour acquérir les données, la technique d'analyse tomographique ou par coupes, est employée en soumettant le patient au balayage d'un faisceau de rayons X [4] [5].

[2] CT: computed tomography.

1.3.1 Fonctionnement

Dans les appareils modernes, l'émetteur de rayons X (tube à rayons X) effectue une rotation autour du patient en même temps que les récepteurs situés en face, qui ont pour fonction de mesurer l'intensité des rayons après qu'ils aient été partiellement absorbés durant leur passage à travers le corps. Les données obtenues sont ensuite traitées par ordinateur, ce qui permet de recomposer des vues en coupes bidimensionnelles puis des vues en trois dimensions.

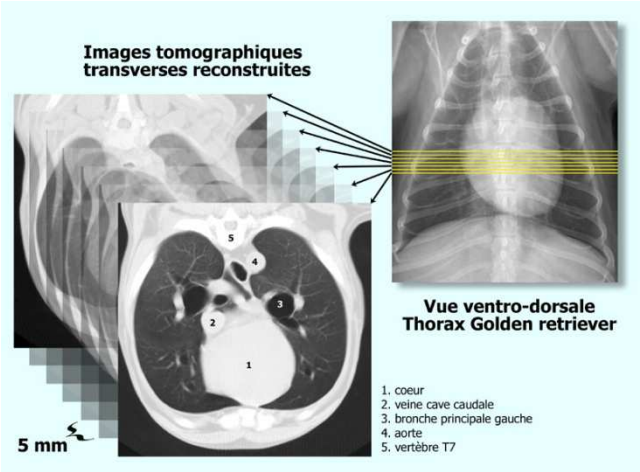


Figure 8 - Exemple de coupes tomographiques

On peut faire ressortir le contraste de certains tissus en injectant un produit dit « de contraste » (un dérivé de l'iode) qui a la propriété de fortement absorber les rayons X et donc de rendre très visibles les tissus où ce produit est présent (qui apparaissent alors hyperdenses, c'est-à-dire plus « blancs » sur l'image qui est en niveaux de gris).

Figure 9 – Balayage de faisceaux de rayons X

Grâce aux tomodensitomètres multi-détecteurs (ou multi-barrettes) à acquisition spiralée (déplacement lent de la table d'examen durant l'acquisition), on obtient depuis les années 1990 une exploration très précise d'un large volume du corps humain pour un temps d'acquisition de quelques secondes.

1.3.2 L'échelle de Hounsfield

Hounsfield a créé une échelle qui porte son nom. Sur cette échelle, il a défini plus de 2000 coefficients de gris pouvant se trouver dans le corps humain. Ces coefficients correspondent à certaines matières telles que l'air, l'eau, les muscles, les os, etc. [5]

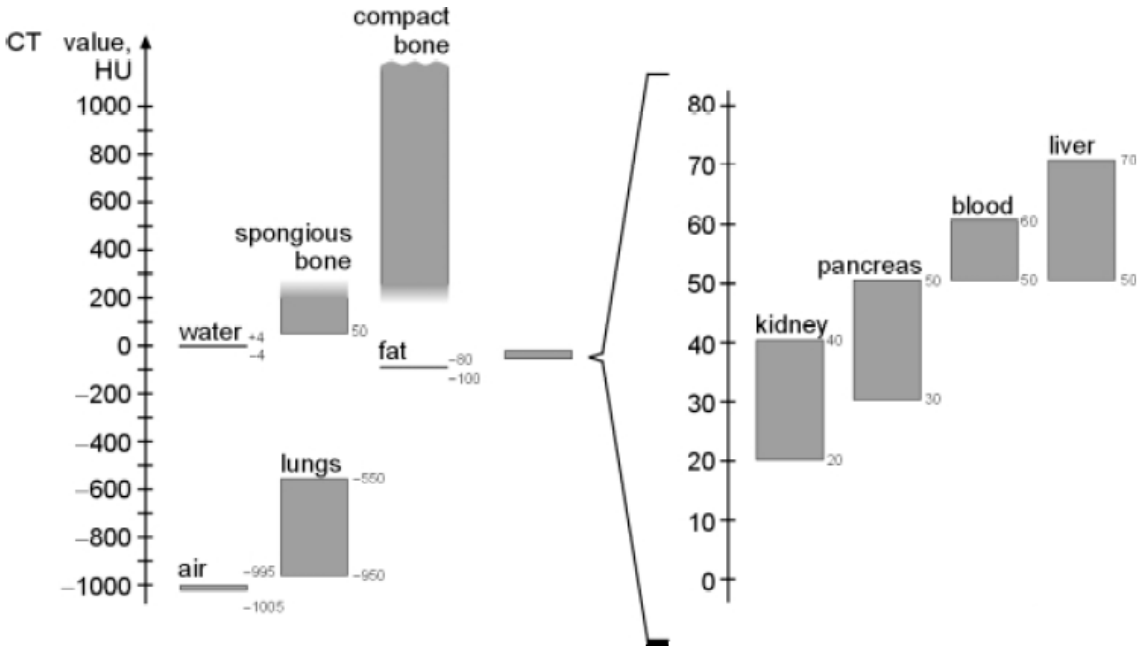


Figure 10 - Échelle de Hounsfield

Hounsfield propose une méthode de visualisation par fenêtrage qui permet de centrer l'étude sur une portion précise de l'échelle, permettant ainsi de ne s'intéresser qu'à certains organes. Il prend comme point de repère l'eau et l'air. Le coefficient de l'eau est de 0 UH[3] et celui de l'air et de -1000 UH.

De manière générale, on distingue trois zones dans l'échelle :

- Les structures peu denses qui se situent dans la partie négative de l'échelle ;
- Les tissus dits mous qui se retrouvent de l'intervalle de -100 à 100 UH ;
- Les structures denses qui s'étendent de +100 à plusieurs milliers d'UH.

Bien que cette échelle offre une grande précision dans les niveaux de gris, l'œil humain n'est capable de discerner qu'une vingtaine de degrés d'atténuations. Si une échelle de gris est répartie sur l'ensemble de l'échelle de Hounsfield, les tissus mous ne sont pas discernés par l'œil par manque de contraste. C'est pour cette raison que la méthode de fenêtrage est utilisée. En effet, elle permet d'étudier avec un bon contraste les tissus souhaités.

Une fenêtre est caractérisée par :

- Son niveau : c'est le centre de la fenêtre, il s'approche de la valeur Hounsfield de la structure étudiée ;

[3] UH : Unité de Hounsfield.

- Sa largeur : elle représente le pouvoir discriminant de la fenêtre. Elle doit être étendue pour l'étude d'un maximum de structures ou étroite pour différencier des structures de densités très voisines.

1.4 Images Utilisées

Les images utilisées tout au long de l'élaboration de ce travail ont été fournies par un examen de type CT-scan. Lors de l'examen, les patients doivent subir une injection d'un produit de contraste à base d'iode afin de faire ressortir les frontières du cartilage. Ces frontières sont repérées par une intensité de couleur fort élevée équivalente à celle de l'os compact.

Chapitre 2 – Objectifs

Le premier chapitre nous a permis de constater les problèmes que peut rencontrer le cartilage de l'articulation du genou ainsi que les différents traitements possibles.

Au vu de la lourdeur des traitements possibles dans les cas les plus atteints par l'arthrose, il semble nécessaire de pouvoir déceler au plus tôt les lésions du cartilage. Ce problème a déjà été abordé par Sébastien Wilfart [6]. Au terme de son mémoire, il proposait une application de segmentation du cartilage du genou.

2.1 L'application

L'application développée par Sébastien Wilfart lors de son stage et mémoire dans le domaine de la segmentation du cartilage du genou offre un bon nombre de fonctionnalités. En effet, l'application permet de procéder à la segmentation du cartilage en utilisant un algorithme de croissance de région (Voir chapitre 3), ainsi que des outils de correction et de visualisation. Par manque de temps, l'application n'a pas pu être dotée d'une interface graphique poussée. Elle s'est donc développée comme une application en ligne de commande.

Bien que les résultats obtenus avec l'application soient positifs, certaines limitations ont été rencontrées. C'est dans ce cadre que s'inscrit ce mémoire. Nous nous positionnons donc dans la continuité du mémoire de Sébastien afin de perfectionner l'application qu'il a offert.

2.1.1 Limitations

Lors de l'utilisation de l'application par le personnel de l'hôpital de Mont-Godinne, nous avons constaté une difficulté de prise en main de l'application. Le fait de combiner des fenêtres demandant l'utilisation du clavier et d'autres, de la souris, rend l'utilisation de l'application ardue.

De plus, il a été noté que l'application souffrait de nombreux arrêts non-programmés en plus d'une utilisation de mémoire excessive. Au fur et à mesure de son exécution, l'application requérait de plus en plus de mémoire sans sembler la libérer par la suite. Cet usage excessif finissait alors par atteindre un point tel que toute la mémoire disponible était occupée, si bien que l'application s'arrêtait.

Sébastien lui-même avait repéré un problème dans l'affichage. Un décalage entre l'image de base et la superposition d'un calque représentant le cartilage. Ce décalage pouvait, lorsque la personne utilisant l'application n'était pas au courant de l'existence du décalage, induire en erreur. En effet, l'utilisateur pense avoir mal configuré l'application ou avoir mal procédé à la sélection de la zone.

Enfin, nous pouvons citer la limitation des résultats en eux-mêmes. Ces derniers ont tendance à sous évaluer le volume du cartilage. Après avoir évalué ses résultats, Sébastien a montré qu'ils ne donnaient pas le même volume que les experts mais un volume plus petit bien que très similaire à celui des expert. L'application ne fournissait donc qu'une idée générale du volume de cartilage.

2.2 Objectifs

Les attentes de monsieur Meurisse étaient tout d'abord liées à l'utilisation de l'application. Il fallait la rendre la plus utilisable possible. Dans cette problématique, une interface graphique, en plus d'une amélioration de la stabilité de l'application, était demandée. Dans le même courant, la suppression du décalage entre l'image et le calque était visée. Cette première étape visait donc à stabiliser l'application ainsi qu'à la rendre plus facile à utiliser.

Dans un deuxième temps, monsieur Meurisse demandait l'ajout d'outils supplémentaires afin de faciliter certaines étapes et de permettre à l'utilisateur de gagner un certain temps. Dans ces outils figuraient un outil de zoom sur l'image et la suppression d'une graine.

Après ces deux étapes visant la facilité de l'utilisateur, les objectifs suivants étaient liés à l'amélioration des résultats. La limitation constatée par Sébastien était due à l'algorithme utilisé qui ne s'approche pas suffisamment des bords du cartilage. Il nous fallait donc trouver une méthode de segmentation qui se rapproche plus des bords afin que les résultats ressemblent davantage à ceux des experts. Dans ce troisième objectif, plusieurs possibilités s'offraient à nous :

- L'utilisation d'un autre algorithme seul ;
- La combinaison avec un autre algorithme ;
- Le passage à une segmentation en 3D.

La première possibilité n'a pas été prise en compte car Sébastien avait déjà exploré cette voie. Dans les chapitres suivants nous verrons comment nous avons utilisé les deux autres possibilités.

Chapitre 3 – Matériel et méthode

Ce chapitre vise à présenter les méthodes et matériaux utilisés afin d'atteindre les objectifs délimités au chapitre précédent. Dans un premier temps les concepts de l'imagerie médicale ainsi que les normes utilisées dans ce milieu seront présentées avant d'aborder brièvement les formats d'encodage des images ainsi que les moyens disponibles (logiciels, librairies informatiques, etc.) traitant les données sous ces formats.

3.1 Concepts d'imagerie

3.1.1 Caractéristiques

Une image médicale se doit d'être précise afin de porter un diagnostique de façon certaine. Une approximation dans l'image peut avoir de graves conséquences. Afin d'éviter tout problème, une image médicale est toujours accompagnée d'informations permettant de faire la correspondance avec le monde réel.

Ces informations permettent de connaître l'emplacement exact des objets représentés sur l'image ainsi que leur taille (ou volume).

Dans le cas d'appareil de type CT, une image est en faite un volume. En effet, un examen de type CT produit un volume correspondant à l'organe observé par une série de coupes. Ces coupes ont donc une certaine épaisseur et sont donc des images en trois dimensions. Toutes les images CT utilisent un système d'axes orthonormés. On peut donc voir le volume d'une image comme un parallélépipède rectangle dont chaque coté est parallèle à un axe.

Afin de faire la correspondance avec le monde réel, toute image est caractérisée par :

- Une origine : l'origine est le point de départ de la région d'intérêt. Il s'agit de l'un des coins du parallélépipède représentant l'image. Elle est enregistrée sous la forme de coordonnées relatives aux axes orthonormés.
- Un « spacing » : le « spacing » définit la taille d'un pixel (en trois dimensions on parle de voxel) dans le monde réel. Le « spacing » exprime la taille du pixel en millimètre sur chaque axe. Ainsi un voxel n'est pas forcément représenté par un cube dans l'espace, il se peut que se soit un parallélépipède. Plus les valeurs données par le « spacing » (donc la taille de pixel) sont petites, plus la précision est élevée, et inversement.
- Le nombre de pixels : cette information est donnée pour chaque axe. Ces valeurs varient en fonction de la taille de la région d'intérêt et la taille des pixels.
- La matrice de pixel : cette matrice contient les valeurs d'intensité de chaque pixel de l'image.

3.1.2 Cropping

Le « cropping » en imagerie médicale est un recadrage de l'image. Cette technique permet d'extraire une partie de l'image de base afin de ne garder que la région désirée. Lorsque le « cropping » est appliqué sur un volume, c'est un sous-volume qui est extrait.

Pour effectuer un « cropping », il faut définir le nouveau point d'origine (en fournissant ses coordonnées) et le nombre de pixels sur chaque axe.

3.1.3 Resampling

Le « resampling », ou échantillonnage, est la méthode qui permet de créer une nouvelle version d'une image avec une hauteur et/ou largeur différente. Augmenter la taille de l'image de base se dit « Upsampling » tandis que réduire la taille de l'image se dit « Downsampling ».

Quand une image est « Upsampled », son nombre de pixels augmente. Toutefois ce procédé ne crée pas de nouveaux détails non présents dans l'image de base. La quantité d'informations diminue lorsque le nombre de pixels augmente. À l'inverse, quand une image est « Downsampled », son nombre de pixels diminue. Une partie de l'information contenue dans l'image originelle doit être écartée afin que l'image puisse être réduite.

Figure 11 - À gauche, l'image de base ; À droite, l'image ré-échantillonnée

Cette méthode a pour conséquence, après l'application successive d'un « Downsampling » et d'un « Upsampling », la perte de l'image de base. Une certaine quantité d'information a été perdue dans le procédé.

3.1.4 Recalage

Le recalage est une technique qui consiste en la mise en correspondance d'images, ceci afin de pouvoir comparer ou combiner leurs informations respectives. Cette mise en correspondance se fait par la recherche d'une transformation géométrique permettant de passer d'une image, dite « mouvante », à une autre, dite « fixe ».

Le recalage est utilisé en imagerie médicale où ses applications sont nombreuses. Il permet notamment de fusionner plusieurs images d'un même patient, ceci par exemple afin de pouvoir exploiter les informations fournies dans différentes modalités comme l'imagerie scanner, l'imagerie par résonance magnétique, ... Mais il peut également être utilisé pour l'étude de l'évolution au cours du temps d'un patient.

Le recalage est dit monomodal lorsque deux images de la même modalité sont recalées. Lorsque deux images de modalités différentes sont recalées, on parle alors de recalage multimodal.

Le recalage monomodal permet de mettre en correspondance les informations de plusieurs patients, ce qui permet de créer des modèles anatomiques de références appelés atlas.

3.2 La norme DICOM

Les images acquises par les équipements PET et CT, sont encapsulées dans des fichiers servant de conteneurs. Ces fichiers respectent la norme DICOM (Digital Imaging and Communications in Medicine). En plus de contenir les images, ces conteneurs détiennent de nombreuses informations supplémentaires liées au patient qui a subi l'examen (nom, prénom, âge, sexe, etc.), les caractéristiques de l'image, les caractéristiques de la machine (marque, modèle), etc [6].

3.2.1 Origines

Suite à l'accroissement des appareils tomographiques utilisant des formats de données différents, l'ACR (American College of Radiology) et la NEMA (National Electric Manufacturers Association) reconnaissent la nécessité d'utiliser une méthode standard pour le transfert des images et des informations qui y sont liées. En 1983, ils se réunissent pour commencer le développement du standard. Ce dernier aura pour but de :

- Promouvoir la communication des informations des images médicales sans prendre en compte l'appareil utilisé pour leur acquisition ;
- Faciliter le développement et l'expansion du stockage d'images et des systèmes de communication (pouvant s'interfacer avec d'autres systèmes du milieu médical) ;
- Permettre la création de bases de données de diagnostiques pouvant être interrogées par une grande variété d'appareils géographiquement éloignés.

En 1985, la norme DICOM est créée.

3.2.2 Objectifs

Comme les origines de la norme le montrent, l'objectif est de faciliter les transferts d'images entre les machines de différents constructeurs. Avant son apparition, les formats de données utilisés par les différents constructeurs étaient incompatibles et entrainaient des coûts supplémentaires dus à l'achat de logiciels ou à des pertes de temps que des manipulations supplémentaires nécessitaient.

Par ailleurs, le suivi médical des patients, surtout en cas de pathologie lourde nécessitant souvent le transfert d'un établissement de santé à un autre en fonction des moyens et compétences disponibles, a directement bénéficié de l'instauration de cette norme. Les images au format DICOM accompagnant les dossiers médicaux sont lisibles sur tout matériel informatique utilisant la norme.

3.2.3 Le format

Comme nous l'avons dit, les fichiers DICOM renferment beaucoup d'informations. Ces informations sont structurées sous la forme d'une suite de champs. Donc, une image acquise lors d'un examen médical ne remplit qu'un champ particulier d'un fichier DICOM. Un champ est caractérisé par :

- **Une étiquette (tag) :** C'est une clé qui permet d'identifier le champ. Le champ est composé de deux parties constituées chacune de quatre chiffres (ex. : [0010|0030] représente le champ de la date de naissance du patient). La première partie représente le groupe d'information (dans l'exemple, le groupe 0010 désigne toutes les données concernant le patient) et la deuxième représente le numéro de l'élément (dans notre exemple, 0030 désigne la date de naissance du patient). À cette étiquette est lié un nom qui permet de décrire ce que représente l'information (dans notre exemple « patient's birth date »).

- **Une représentation de valeur :** Cette information permet de savoir sous quelle forme a été enregistrée l'information (chaine de caractères, entier, date,…).
- **Une longueur :** Précise la taille maximale de l'information.
- **La valeur :** L'information en elle-même.

	All existing tags
[0002]	File Meta Elements
[0008]	Study information
[0010]	Patient
[0018]	Acquisition Group
[0020]	Relationship Group
[0028]	Image presentation
[0029]	Private
[0032]	Study Schedule Group
[0040]	
[0054]	
[6000]	

[Group, Elément]	Titre	Valeur
[0028:0002]	Samples per Pixel	1
[0028:0004]	Photometric Interpretation	MONOCHROME2
[0028:0010]	Rows	512
[0028:0011]	Columns	512
[0028:0030]	Pixel Spacing	0.267578\0.267578
[0028:0100]	Bits Allocated	16
[0028:0101]	Bits Stored	12
[0028:0102]	High Bit	11
[0028:0103]	Pixel Representation	0
[0028:1050]	Window Center	450\50
[0028:1051]	Window Width	1850\420
[0028:1052]	Rescale Intercept	-1024
[0028:1053]	Rescale Slope	1
[0028:1055]	Window Center _Width Explanatio	WINDOW1\WINDOW2

Figure 12 - Exemple de champs DICOM

3.2.4 Champs utiles

La connaissance de la norme DICOM est importante dans le traitement d'images médicales. En effet, il est possible de retrouver les informations nécessaires sur l'encodage des images, l'appartenance à une série d'images,… Voici une liste non exhaustive de champs qui ont été utiles.

TAG	NOM	DESCRIPTION
[0028|0010]	Rows	Le nombre de pixels sur l'axe Y
[0028|0011]	Columns	Le nombre de pixels sur l'axe X
[0054|0081]	Number of slices	Le nombre de coupes dans la série d'images représentant le volume entier
[0028|0030]	Pixel spacing	Donne la taille d'un pixel, dans l'espace, en millimètre
[0018|0050]	Slice thickness	Donne l'épaisseur en millimètre de la coupe

3.3 Le Format MetaImage

Le format DICOM, malgré ses nombreuses qualités, n'est pas aisé à utiliser pour le traitement de volumes. En effet, chaque fichier représente une coupe d'un volume et de nombreux fichiers doivent être manipulés pour traiter le volume. Pour pallier à ce problème, nous avons décidé de transformer nos séries de fichiers DICOM en fichiers au format MetaImage, format défini par ITK. Ce format a l'avantage de représenter le volume dans son ensemble, facilitant ainsi la gestion de fichiers [7].

3.3.1 Les fichiers

Les images enregistrées sous ce format sont composées de deux fichiers. Le premier, « *monfichier.mhd* », représente le « *Meta header data* », un fichier d'en-tête. Il contient certaines informations sur l'image. Au minimum, il prend la forme de l'image suivante :

```
1  NDims = 3
2  BinaryDataByteOrderMSB = False
3  Offset = 7.1401 -255.352 215.089
4  ElementSpacing = 0.300781 0.300781 1.50562
5  DimSize = 512 512 64
6  ElementType = MET_SHORT
7  ElementDataFile = monfichier.raw
```

Figure 13 - En-tête minimal de fichiers MetaImage

Ces données sont reprises automatiquement des champs des fichiers DICOM représentant l'image. Voici la signification de ces champs :

- **NDims :** Ce champ représente le nombre de dimensions de l'image. Dans notre exemple, il s'agit d'un volume car l'image possède trois dimensions.
- **BinaryDataByteOrderMSB :** Ce champ mentionne si les données sont encodées en « *Little endian* » ou en « *Big endian* ». Il est important de connaître cette information pour bien interpréter les données représentant l'image. En effet, une mauvaise interprétation donnera une image dont les couleurs seront totalement différentes et dont les formes ne seront pas respectées.
- **Offset :** Ce champ indique la position de l'origine de l'image par rapport aux axes en utilisant comme unité de mesure le millimètre.
- **ElementSpacing :** Ce champ indique la taille, en millimètre, d'un pixel, dans toutes les directions.
- **DimSize :** Ce champ s'agit de la taille de l'image en pixel dans les différentes dimensions.
- **ElementType :** Ce champ indique la façon dont les pixels sont encodés. Dans notre exemple, il s'agit de « *SHORT* », les valeurs sont donc comprises entre -32768 et 32767.
- **ElementDataFile :** Ce dernier champ (il doit toujours se trouver en fin du fichier d'en-tête) donne le nom du fichier contenant les données des pixels de l'image. Il faut impérativement que ce fichier se trouve dans le même répertoire que le fichier d'en-tête.

Le second fichier, « *monfichier.raw* », contient les données binaires brutes des pixels. Les valeurs des pixels sont inscrites dans le fichier à la suite l'une de l'autre. Pour interpréter les données de façon correcte, le fichier « *.mhd* » doit être bien défini.

Il est possible de regrouper le fichier d'en-tête et celui des données brutes dans un seul fichier « *.mha* ». Cette procédure a l'avantage de compresser les données et donc de gagner en espace disque. Mais cela rend plus compliquée la lecture des caractéristiques de l'image.

3.4 Bibliothèques d'imagerie et librairies

3.4.1 ITK

ITK (Insight Segmentation and Registration Toolkit) est un framework C++ open source (licence BSD[4]) qui propose aux développeurs un ensemble d'outils logiciels de traitement d'images.

[4] Licence libre utilisée pour la distribution de logiciels. Elle permet de réutiliser tout ou une partie du logiciel sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire.

De plus, ce framework est multiplateforme. Il propose un ensemble d'algorithmes implémentés de segmentation, de filtrage et de déformation d'images [8].

ITK a été indispensable pour mener à bien les objectifs fixés. En effet, nous avions besoin d'algorithmes de segmentation et de déformation d'images.

ITK fut créé à la demande de l'institut américain de la santé. Ce dernier a signé, en 1999, un contrat de financement de trois ans pour le développement d'un framework facilitant les manipulations d'images afin de réduire les coûts et les temps de production de logiciels médicaux.

3.4.2 VTK

VTK (Visualization ToolKit) est un framework C++ open source (licence BSD) conçu pour faciliter la visualisation d'images. Il permet notamment la visualisation d'objets 3D et met à disposition des outils d'interaction avec les images ou objets 3D [9].

VTK a été utile pour la visualisation des images sous format DICOM ainsi que les images 3D au format MetaImage.

3.4.3 Qt

Qt[5] est un framework d'applications multiplateforme permettant aux développeurs de logiciels commerciaux et open source de créer des applications et des interfaces utilisateur innovantes. Qt est basé sur un système de double licence : une licence commerciale pour le développement de logiciels propriétaires et une licence publique générale (« GPL »[6] versions 2 et 3) pour le développement de logiciels libres et open source [10].

Qt possède des mécanismes de liaison avec VTK et ITK, facilitant l'intégration des fenêtres de visualisation de VTK dans les interfaces de Qt. Ce qui explique qu'il a été choisi pour développer notre interface graphique.

[5] Prononcé « *cute* ».
[6] Licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU.

3.4.4 GDCM

GDCM est une librairie C++ open source multiplateforme. Elle permet de lire et écrire les fichiers dans la norme DICOM et de remplir ces fichiers DICOM champ par champ. Cette librairie a été utilisée afin de faire le lien entre les images médicales et les algorithmes fournis par ITK.

3.5 Traitement d'image

3.5.1 Segmentation

Nous allons voir les méthodes de segmentation d'images qui ont été utilisées. D'autres méthodes ont été testées mais rejetées, nous ne les aborderons pas. Dans cette partie, le critère de comparaison entre pixels est la différence de couleur. Dans le cas d'images médicales provenant d'appareils CT, il s'agit d'images en niveaux de gris. Nous parlerons donc de différences de niveaux de gris plutôt que de différences de couleurs.

3.5.1.1 Croissance de région

La méthode de segmentation par croissance de région [11] [12] (« *Region Growing* » en anglais) est une des approches conceptuelles les plus simples. Elle a l'avantage d'être assez rapide mais ne prend pas en compte tout le problème dès le départ. En effet, le voisinage d'un pixel est ajouté dans la même région s'il a un niveau de gris similaire. Il existe plusieurs variantes de cette méthode :

- La première inclut dans la région un pixel adjacent à celle-ci s'il correspond bien au critère de similitude. Dans notre cas, un pixel adjacent à la région segmentée y est ajouté s'il a un bon niveau de gris.
- La seconde n'inclut un pixel dans la région que si tous les pixels de son voisinage direct respectent le critère de similitude. Donc s'ils ont un bon niveau de gris. Cette deuxième variante est plus restrictive quant à la croissance. En effet, elle aura tendance à s'arrêter plus rapidement que la première variante.

La première étape dans les méthodes de croissance de région est de sélectionner un ensemble de pixels de départ, appelé graines. La sélection des pixels de départ se fait par l'utilisateur, il doit bien évidemment sélectionner des pixels respectant le critère de similitude. Ces graines forment la région de départ de la méthode qui va ensuite ajouter les pixels adjacents à la région s'ils respectent le critère. Une fois que la région ne croit plus, la méthode est terminée.

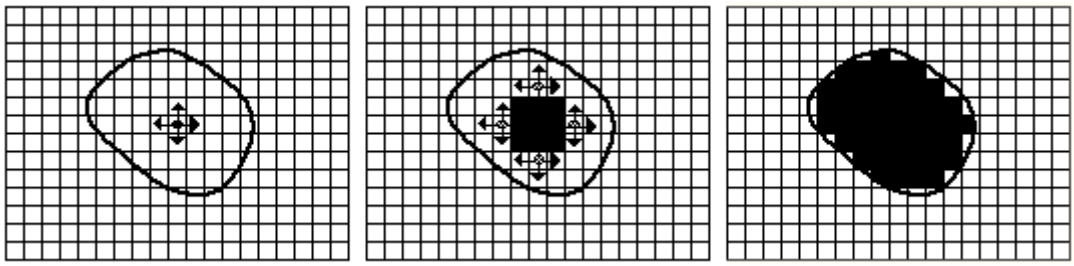


Figure 14 - Fonctionnement de la croissance de région

Comme mentionné ci-dessus, cette méthode ne prend pas en compte toute l'image dès le départ, ce qui en fait une bonne méthode pour segmenter un organe précis et bien délimité. Par

contre si le but est de sélectionner tous les pixels de l'image qui respectent le critère de similitude, alors les méthodes par croissance de région ne sont pas du tout adaptées.

3.5.1.2 Ligne de partage des eaux

La segmentation par ligne de partage des eaux[7] [12] [13] (« *Watershed* » en anglais) désigne une famille de méthodes dans lesquelles la segmentation est envisagée comme un problème géographique. En effet, en associant une valeur altitude à chaque niveau de gris, un relief topographique peut être construit. La méthode simule ensuite l'inondation du relief. Cette méthode n'est que rarement appliquée sur l'image originale. Un filtrage est préalablement appliqué pour lisser l'image et, ensuite, l'image gradient est calculée. C'est sur cette dernière qu'est souvent appliquée la méthode par ligne de partage des eaux car le gradient accentue le relief.

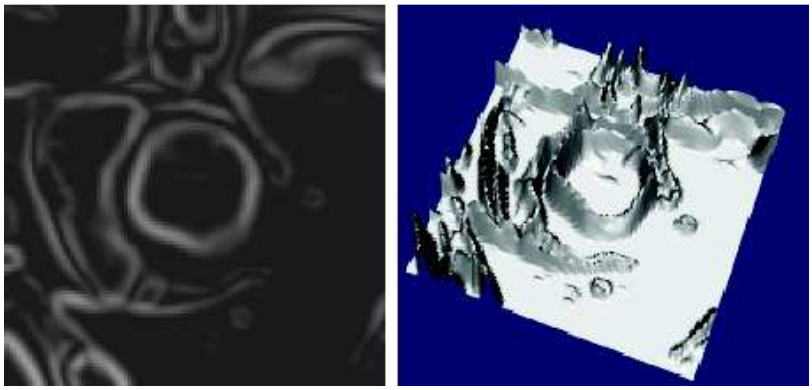


Figure 15 - À gauche, l'image filtrée ; À droite, l'image perçue en « Watershed »

Illustrons le fonctionnement de cette méthode en prenant en compte le relief suivant.

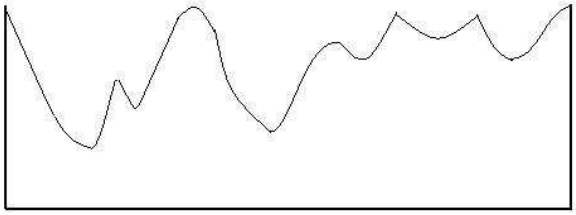


Figure 16 - Exemple de relief

Un minimum local est un point d'où il est impossible de rejoindre un autre point de hauteur inférieure sans passer par des points d'une hauteur supérieure.

[7] Ligne de Partage des Eaux LPE

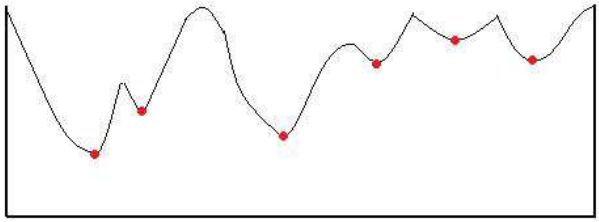


Figure 17 - Minima locaux d'un relief

Un bassin versant est lié au concept de minimum local. En effet, un bassin est une zone du relief qui, si on verse du liquide, amènera ce liquide jusqu'au minimum.

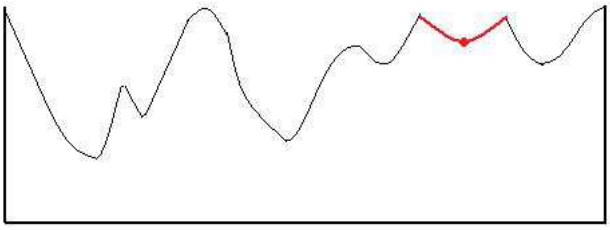
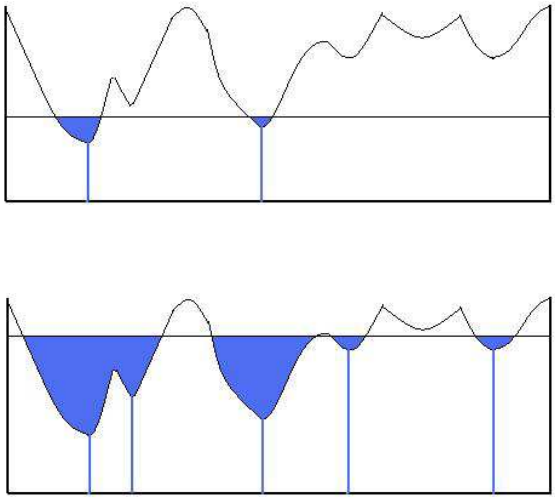


Figure 18 - Exemple de bassin versant

Une ligne de partage des eaux est une ligne qui sépare deux bassins versants telle une digue empêchant l'eau de passer d'un bassin à l'autre. Pour découvrir ces lignes, la technique utilisée est d'inonder progressivement les bassins à partir des minima. Les différents bassins se remplissent ainsi progressivement et la ligne de partage des eaux passe par le point où deux bassins adjacents finissent pas n'en faire plus qu'un.



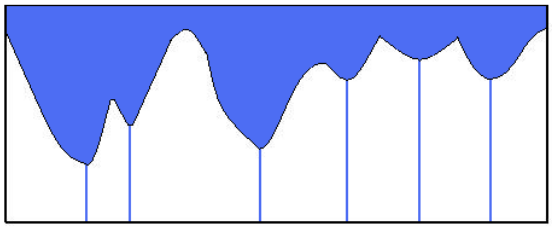


Figure 19 - Illustrations du fonctionnement du LPE

Cette méthode présente l'avantage de prendre en compte toute l'image dés le départ, contrairement aux méthodes de croissance de région. De plus, elle ne génère pas qu'une seule région d'intérêt mais un ensemble de régions. Cet ensemble peut être représenté comme un arbre dont la racine est l'image complète.

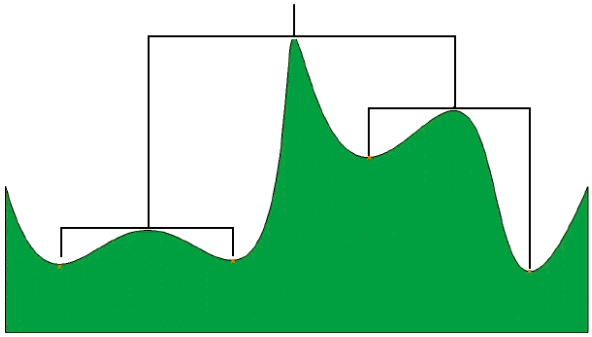


Figure 20 - Exemple d'arbre des régions

En fonction du nombre de régions désiré, la méthode permet de définir la hauteur de l'arbre par l'intermédiaire d'un niveau d'inondation. Ce qui se traduit par la quantité d'eau qui s'accumule dans un bassin. Ce paramètre s'appelle en général « *level* » ou « *flood level* ».

Il faut mentionner que la méthode génèrera une région pour chaque minimum local. Cela engendre une sur-segmentation de l'image. On définira alors une quantité maximale d'eau qu'un bassin peut contenir avant d'être fusionné avec son voisin. Ce paramètre s'appelle « *threshold* ».

3.5.1.3 Coopération d'algorithme

La coopération d'algorithme de segmentation est une technique qui permet d'affiner le résultat obtenu par un seul algorithme. Il existe plusieurs méthodes de coopération, nous en présenterons trois, la coopération séquentielle, la coopération de résultats et la coopération mutuelle [14].

La coopération séquentielle

Le principe général de la coopération séquentielle est que l'un des algorithmes choisi travaille avant le ou les autre(s) pour fournir son résultat en plus de l'image de base. Cela permet d'apporter de l'information supplémentaire au second algorithme. Cette information peut avoir plusieurs formes.

La première forme d'information complémentaire est la définition d'un nouveau critère de segmentation. Dans le cas présenté dans la figure suivante, les contours fournis par le premier

algorithme peuvent être utilisés pour interdire au second de prendre en compte les pixels du contour.

La seconde forme est l'ajustement d'un critère de segmentation. Toujours dans le cas de notre figure, l'analyse d'échantillons de pixels de part et d'autre des contours permettra de rendre le critère d'homogénéité plus adéquat pour l'algorithme de croissances de régions. Mais inversement, le résultat d'une segmentation par régions peut aider à détecter les contours. Par exemple, en utilisant les limites obtenues par la croissance de régions comme point de départ pour l'algorithme de détection de contours.

Ce type de coopération permet aussi de faciliter la pose de graine ou d'éliminer des mauvaises zones dans la segmentation. Cela permet aussi, dans certains cas, de réduire le temps de calcul.

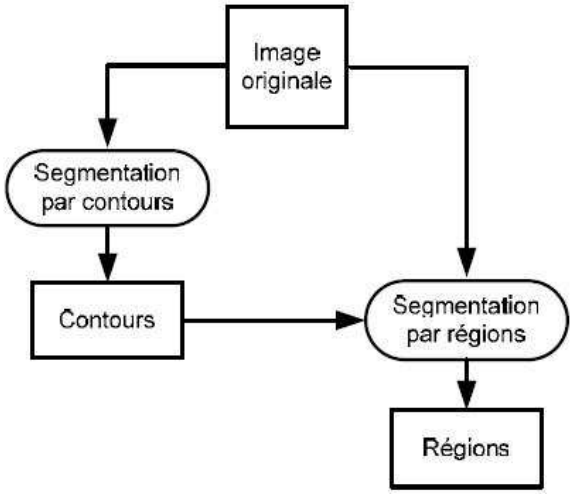


Figure 21 - Segmentation par coopération séquentielle

La coopération de résultats

Dans ce type de coopération, l'idée générale est que les algorithmes sont exécutés de façon indépendante. Les résultats obtenus par les deux algorithmes sont alors fusionnés pour obtenir un meilleur résultat global.

Les deux résultats peuvent simplement être additionnés afin d'obtenir une zone plus importante. Cette technique permet aussi de diviser une grande segmentation d'un algorithme en plusieurs zones plus petites grâce au résultat d'un autre algorithme donnant ainsi une information complémentaire sur l'homogénéité de la région.

Ce type de coopération permet aussi de ne considérer que ce qui est en commun aux différents résultats pour avoir une région plus restreinte mais entrant d'avantage dans les paramètres. La coopération peut contribuer aussi au problème de paramétrage des segmentations. Par exemple, plusieurs segmentations par croissance de régions avec différents paramètres sont réalisées. Pour juger de la meilleure segmentation, les résultats sont comparés à une segmentation par détection de contours. La segmentation dont les limites de régions sont les plus proches des contours est retenue.

Il est possible aussi de mettre en place un système itératif d'ajustement des paramètres. Les itérations sont faites avec des critères de plus en plus tolérants jusqu'à la convergence vers des résultats cohérents et stables. La vérification de la cohérence est basée sur la minimisation de la différence entre les contours et les régions.

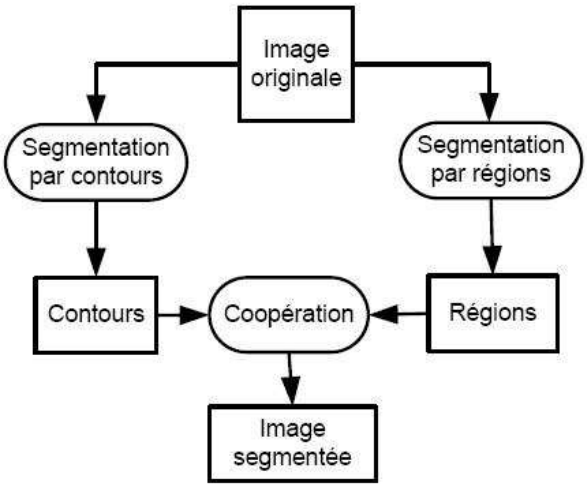


Figure 22 - Segmentation par coopération de résultats

La coopération mutuelle

Le principe de ce dernier type de coopération consiste en une exécution en parallèles des algorithmes qui s'échangent des informations pendant le traitement et permettent ainsi de combler les lacunes dans les paramètres de base. Les décisions prises par les algorithmes sont alors plus sûres. Cette technique est de loin la plus compliquée à mettre en place. Nous ne la développerons pas plus.

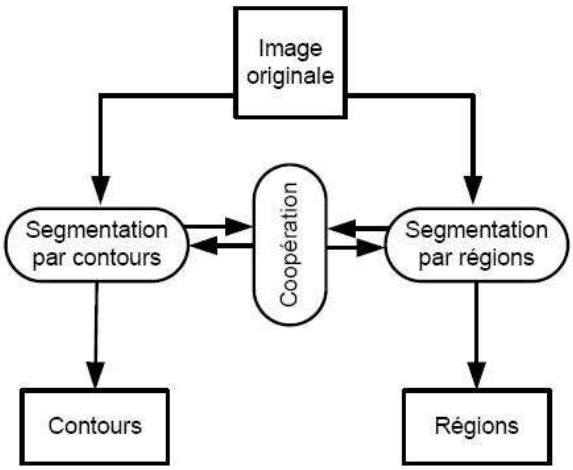


Figure 23 - Segmentation par coopération mutuelle

3.5.1.4 Guidage par atlas

Le guidage d'une segmentation par un atlas n'est pas, à proprement parler, une méthode de segmentation. C'est un processus qui permet d'améliorer les résultats obtenus par une méthode de segmentation. Un atlas est un résultat (ou ensemble de résultats) préalablement obtenu, corrigé et

accepté par une communauté d'experts du domaine. Chaque résultat d'un atlas est constitué de la série d'images médicales de base et du volume segmenté correspondant à cette série.

L'utilisation d'un atlas permet de sélectionner l'ensemble des pixels de départ (graines) sans interaction de l'utilisateur. Il peut être utilisé aussi pour enlever des zones ayant été sélectionnées mais ne faisant pas partie de la région d'intérêt. Une dernière utilisation est de se servir de l'atlas comme d'une barrière pour la segmentation, rendant tout processus de correction ultérieur inutile.

Le choix de l'atlas ou sa construction peut se faire de différentes façons. Nous allons décrire les quatre méthodes les plus connues [15].

Atlas mono-résultat

Dans ce cas, soit l'atlas n'est constitué que d'un seul résultat soit un résultat est sélectionné de manière aléatoire. C'est la méthode la plus simple. En effet, cette technique demande moins d'intelligence dans la préparation et la sélection de l'atlas. Une fois l'atlas sélectionné, il doit être déformé afin que celui-ci ressemble le plus possible à l'objet qui sera segmenté (dans notre cas, au genou dont doit être segmenté le cartilage).

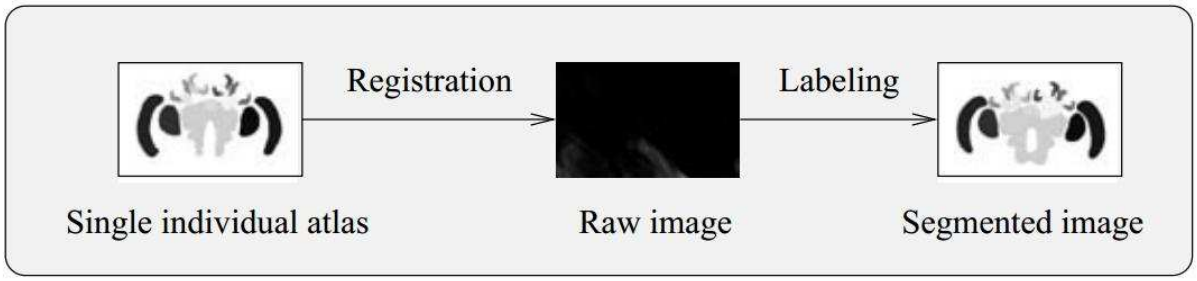


Figure 24 - Atlas comprenant un seul résultat

Atlas le plus ressemblant

Lorsqu'un atlas est constitué de plusieurs résultats, une méthode encore simple est de comparer tous les résultats avec l'objet à segmenter et de ne retenir que le plus ressemblant. Le résultat de l'atlas repris pour la segmentation est ainsi le plus approprié. Pour comparer un atlas avec l'image à segmenter, il existe deux nombres caractéristiques qui décrivent la similitude. Le premier est la valeur finale du critère de recalage, après un recalage (déformation). Le deuxième est l'amplitude de la déformation requise pour mapper les coordonnées de l'image avec celles de l'atlas.

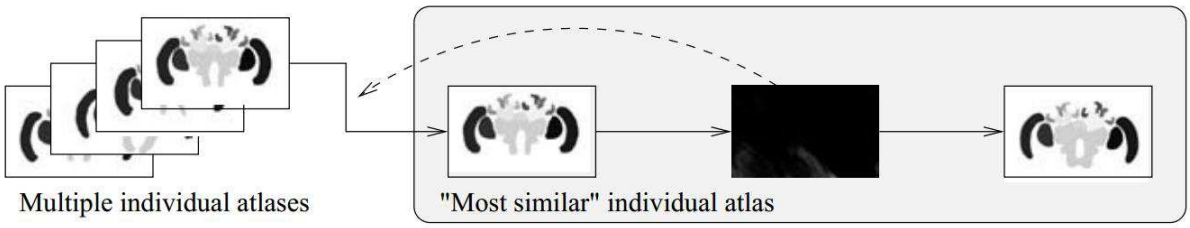


Figure 25 - Résultats de l'atlas le plus ressemblant

Sur base de ces deux concepts, nous allons décrire quatre critères de comparaison :

- **Similarité d'image après transformation affine**[8] **:** L'image de l'atlas qui possède la plus haute similarité avec l'image à segmenter après avoir subi des transformations affines est sélectionnée.
- **Similarité d'image après un recalage non-rigide**[9] **:** L'image de l'atlas qui possède la plus haute similarité avec l'image à segmenter après avoir subi des transformations non-rigide est sélectionnée.
- **Déformation moyenne de l'atlas :** Après un recalage non-rigide, la magnitude de la déformation effectuée est calculée pour chaque pixel. Dans ce cas, l'atlas qui possède la plus petite moyenne est sélectionné.
- **Déformation maximale de l'atlas :** Ce critère est semblable au précédent si ce n'est que l'atlas est utilisé avec la déformation maximale plutôt que la plus petite moyenne. L'idée, ici, est qu'un atlas qui ressemble dans l'ensemble à l'image à segmenter peut ne pas donner de bons résultats pour certaines régions.

Moyenne des atlas

Cette troisième solution essaie de passer outre les cas hors normes en tentant de créer un atlas proche du maximum de personnes différentes. Cela est possible en construisant un atlas qui est la moyenne de nombreux autres atlas.

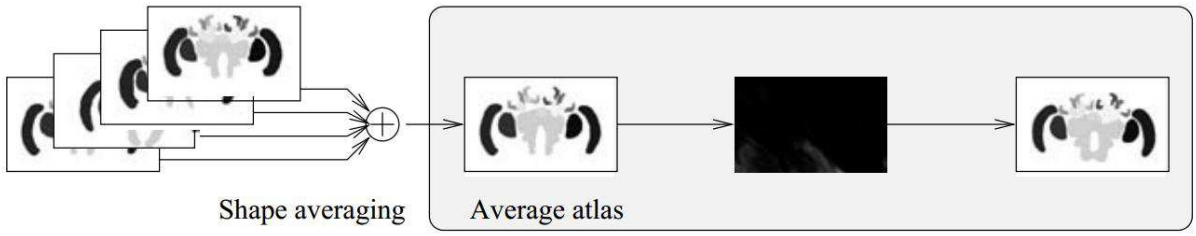


Figure 26 - Moyenne de l'atlas

Une possibilité d'obtenir un tel atlas est de générer un ASM[10]. Un ASM fournit une description statistique d'une population de sujets à l'aide d'une forme moyenne et des principaux modes de variation. La génération d'un ASM requière l'identification de points de repère valables pour toute la population. C'est un processus fastidieux qui est souvent source d'erreurs. D'autres méthodes de création de tels atlas sont basées sur le recalage non-rigide.

Multi-Atlas

En utilisant des atlas différents, on aboutit à des segmentations différentes d'une même image donnée. Il est connu que plusieurs atlas indépendants peuvent être combinés et, ensemble, atteindre une précision supérieure à celle de tous les atlas d'origine pris individuellement. Cette dernière approche, plus compliquée, vise donc à faire travailler l'atlas en complet, voire même plusieurs atlas, afin de produire un résultat plus précis. Pour ce faire elle considère un atlas comme un classificateur. L'entrée du classificateur est une coordonnée dans le domaine de l'image brute. La sortie du classificateur, déterminée en interne par la transformation qui coordonne et regarde

[8] Transformation affine : transformation qui conserve le parallélisme et les ratios de distance entre points et droites
[9] Transformation non-rigide : transformation qui ne conserve pas le parallélisme et les ratios de distance entre points et droites.
[10] ASM (« *Active Same Model* »).

l'étiquette dans l'atlas à l'endroit transformé, est l'étiquette que le classificateur attribue à la coordonnée de l'image à segmenter.

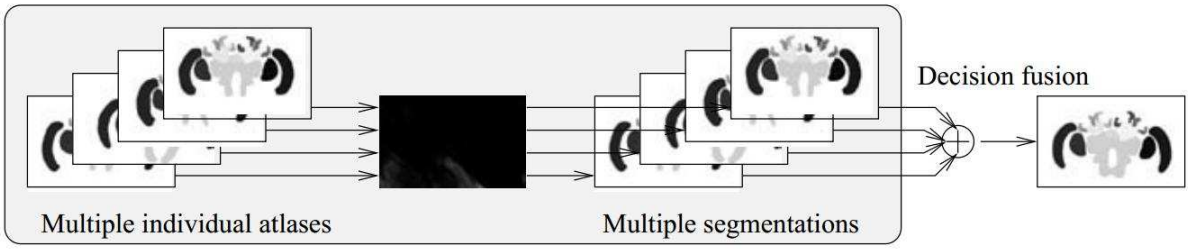


Figure 27 - Atlas multi-résultat avec décision de fusion

3.5.2 Déformation

3.5.2.1 Diffeomorphic Demons

L'algorithme « Diffeomorphic Demons » est l'algorithme de recalage inclus dans ITK [16] [17] que nous avons utilisé dans le processus de segmentation guidé par atlas.

Cette implémentation utilise l'algorithme démon de J.-P. Thirion [18]. Cet algorithme porte le nom de « *démon* » car il se base sur le concept des démons de Maxwell, introduit au 19ème siècle pour illustrer un paradoxe de la thermodynamique.

3.5.2.2 Algorithme Démons de Thirion

J.-P. Thirion propose une vision particulière du recalage d'image. Il le compare à un processus de diffusion d'une image vers la seconde. De cette façon, les objets apparaissant sur l'image dite fixe retrouvent leurs correspondants sur l'image dite mouvante. Toutefois le passage d'une image vers l'autre se fait par l'application d'une diffusion ou déformation.

Pour cerner le fonctionnement de l'algorithme, il faut percevoir les deux images comme semi-perméables et superposées l'une sur l'autre, comme sur la figure ci-dessous.

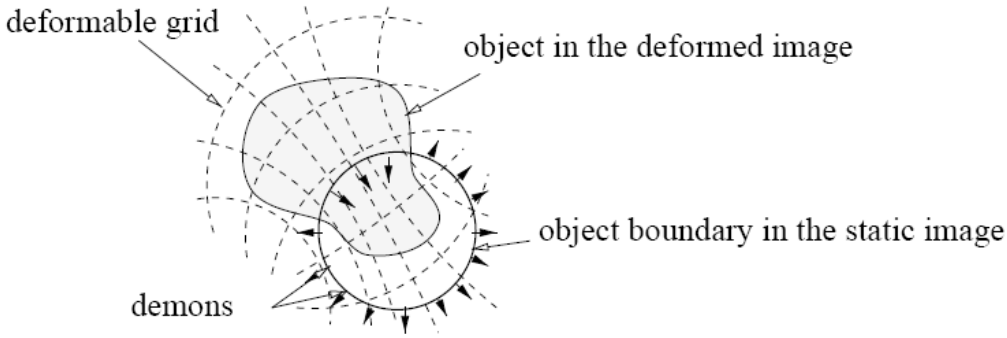


Figure 28 - Modèle de diffusion : une image déformée, considérée comme une grille déformable, est diffusée au travers des contours des objets de l'image statique par l'action des démons

Dans cet algorithme, il faut considérer l'image mouvante comme une grille déformable dont chaque sommet est une particule qui peut être étiquetée comme :

- **Intérieur :** si la position de la particule se trouve à l'intérieur du contour (membrane) de l'objet.

- **Extérieur :** Si la position de la particule se trouve à l'extérieur du contour (membrane) de l'objet.

Les contours des objets sont considérés comme des membranes sur lesquelles il est possible de positionner des objets que Thirion appel démons.

Pour lui, un démon est un effecteur situé en un point *P* de la frontière d'un objet *O* sur l'image fixe. Son rôle est de pousser localement le modèle *M* à l'intérieur de *O* si le point correspondant de *M* est étiqueté « Intérieur » et à l'extérieur de *O* s'il est étiqueté « Extérieur ».

Il est donc possible de déterminer, pour chaque démon, un vecteur perpendiculaire à la membrane et orienté soit vers l'intérieur de l'objet soit vers l'extérieur et ayant pour but de traduire l'action du démon.

Illustrons le fonctionnement de la méthode sur l'exemple suivant :

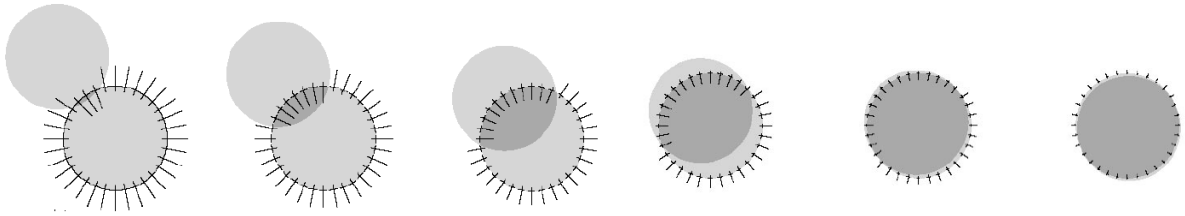


Figure 29 - Plusieurs itérations de modèles de diffusion de type démon

Un ensemble de démons est placé sur le contour du disque de l'image fixe. À chaque démon est attribué un vecteur dont la force est orientée de l'intérieur vers l'extérieur du disque lorsque le point du modèle correspondant est « Extérieur ». Si le point est classé « Intérieur », la polarité est centrale. La méthode proposée procède par étape. À chaque itération, le mouvement engendré par les démons est appliqué au modèle. Afin de garantir une convergence, la magnitude des forces est diminuée à chaque itération.

Remarquons que si les objets des deux images n'ont pas d'intersection comme sur la figure suivante, la méthode est inefficace.

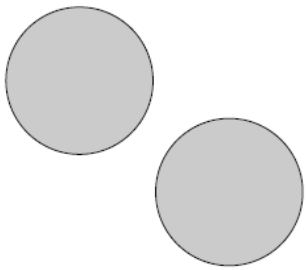


Figure 30 - Cas problématique pour l'algorithme des démons de Thirion

Déroulement de l'algorithme

La première étape de l'algorithme est de sélectionner l'emplacement des démons sur l'image fixe. Il existe plusieurs stratégies pour cela. La première consiste à choisir tous les points de l'image.

La deuxième ne prend que les contours des objets présents sur l'image. La dernière choisit des points extraits grâce à des méthodes de détection avancées.

La seconde étape est itérative et se divise en deux sous-étapes. La première calcule individuellement la force de chaque démon, qui dépend de la direction et du sens du vecteur qui lui est associé. La seconde étape génère une transformation générale de l'image en considérant toutes les forces et l'applique sur l'image mouvante.

Il existe plusieurs versions de cet algorithme qui diffèrent par la détermination des démons, le type de déformation autorisée, les méthodes d'interpolation des forces pour la génération des transformations, etc.

Algorithme démons difféomorphique

L'algorithme de Thirion a été optimisé par une équipe de chercheurs [19] qui en a proposé une implémentation. Il s'agit de la version difféomorphique[11] qui se trouve dans les algorithmes offerts par ITK. C'est la version la plus performante à ce jour.

Dans cette version, l'algorithme utilise un espace de difféomorphisme. Cet espace ne forme pas un espace de vecteurs, mais des groupes de Lie[12]. De ce fait l'optimisation se fait sur ces groupes de Lie. Cela nécessite l'utilisation d'optimisations géométriques qui ont l'avantage :

- De conserver les structures géométriques des groupes ;
- D'utiliser des routines d'optimisation sans contrainte ;
- De garantir la même convergence que les méthodes de Newton classiques pour les espaces de vecteurs.

Option de l'algorithme

L'algorithme offert par ITK offre deux choix intéressants, le premier est le nombre de niveaux de résolution et le second est le nombre d'itérations dans chaque niveau.

Le principe des niveaux de résolution est assez simple. Le premier niveau travaille sur l'image dont la résolution a été fortement diminuée, les niveaux suivants travaillent la même image dont la résolution est à chaque fois plus élevée. Le dernier niveau travaille sur l'image à pleine résolution. Cela permet de trouver une approximation de déformation plus rapidement vu le nombre réduit de pixels lors des premiers niveaux (ce qui engendre une réduction de calculs). On itère ensuite sur des images de meilleures résolutions afin d'améliorer la précision. L'algorithme fourni dans ITK autorise jusqu'à six niveaux de résolution. Nous avons décidé, suite au conseil de la communauté d'ITK, de limiter ce nombre à quatre.

L'algorithme propose également de fournir pour chaque niveau un nombre d'itération afin de définir le niveau de finesse de la déformation.

[11] Un difféomorphisme est un isomorphisme dans la catégorie des variétés différentielles : c'est une bijection différenciable d'une variété dans une autre, dont la bijection réciproque est aussi différentiable.
[12] En mathématiques, un groupe de Lie est un groupe « lisse », qui possède une structure différentiable pour laquelle les opérations de groupe – multiplication et inversion – sont différentiables.

3.6 Outils de développement et de visualisation

3.6.1 ParaView

ParaView est une application open source multiplateforme d'analyse et de visualisation d'image. Il permet de visualiser des objets 3D ainsi que des champs de vecteurs. Cette application a l'avantage de supporter les formats DICOM et MetaImage [20].

Elle a été implémentée par l'équipe de développement d'ITK afin d'avoir une application supportant tous les formats utilisés par ITK.

3.6.2 Telemis

Telemis est une solution globale et évolutive de gestion d'images médicales destinée aux hôpitaux et cliniques ou aux cabinets privés. Telemis a été développé en java par la société Telemis S.A.; il comporte plusieurs modules (plugins) qui permettent de contrôler les différents stades de l'utilisation d'images médicales numériques depuis la production jusqu'à l'archivage. Les images peuvent être partagées entre les départements produisant les images et les autres services. Le produit permet un accès rapide, facile et sécurisé à toutes les images [21].

La solution mise en place à Mont-Godinne utilise Telemis-Medical (TM) qui est la version de Telemis pour hôpitaux. Ce logiciel comporte quatre étapes dans le cycle de vie d'une image médicale, comme le montre l'image suivante :

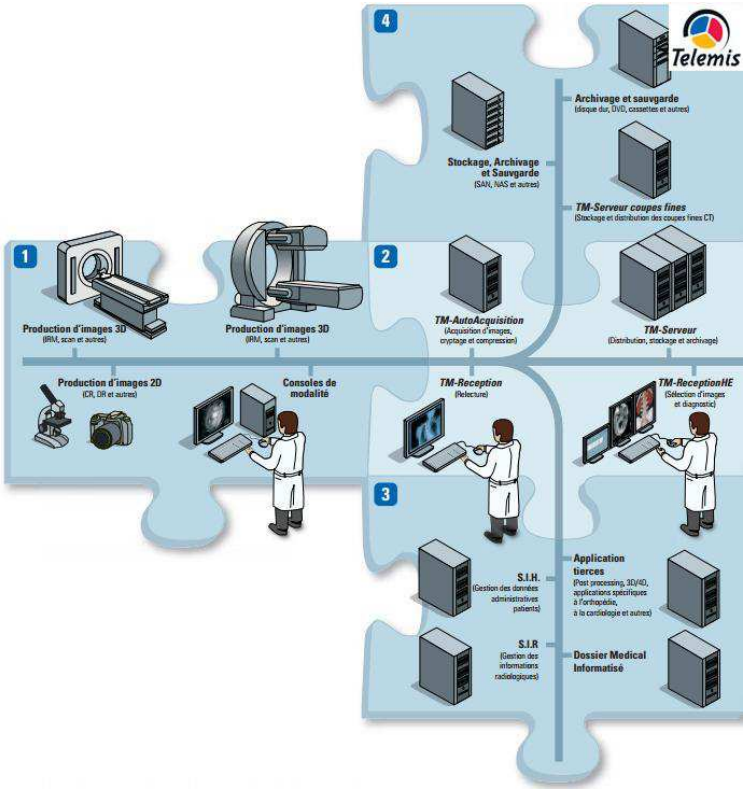


Figure 31 - Gestion d'imagerie médicale par Telemis

1- L'acquisition de l'image directement après sa production par les équipements médicaux ;

2- La distribution de l'image pour la consultation des médecins suivant le patient ;
3- L'intégration de l'image dans le dossier du patient au sein de l'hôpital ;
4- L'archivage de l'image pour sa conservation.

Telemis est utilisé quotidiennement au sein des cliniques universitaires de Mont-Godinne. Il a l'avantage de permettre l'ajout de plugins. Nous avons donc développé notre application comme un plugin venant s'ajouter à Telemis.

3.6.3 CMake

CMake est un système open source qui gère la compilation des applications pour que celles-ci soient indépendantes du système d'exploitation. Bien qu'il gère la compilation, CMake travaille conjointement avec le compilateur natif du système d'exploitation. Il faut donc avoir préalablement installé un compilateur du langage voulu pour utiliser le logiciel CMake [22].

L'utilisation de CMake pour la compilation d'un projet demande la création d'un fichier de configuration (« *CMakeLists.txt* ») utilisant une syntaxe particulière. Dans ce fichier, nous devons spécifier les liens vers les librairies utilisées dans le projet. Cela permet ensuite à CMake de générer les makefiles[13] adaptés à l'environnement de compilation du développeur.

Il est possible de créer manuellement les fichiers makefile, mais dans un projet comme le nôtre, de tels fichiers sont très volumineux et très nombreux. L'utilisation de CMake est donc indispensable.

3.6.4 MinGW

MinGW (Minimalist GNU for Windows) est un environnement de développement minimal adapté de GNU pour Windows. MinGW fournit un ensemble de logiciel open source pour le développement d'application sur les plateformes Windows (dont un compilateur C++) [23].

MinGW nous a fourni le compilateur qui nous a permis de développer une application C++.

3.6.5 MSYS

MSYS (Minimal System) est un système d'interprétation de lignes de commandes. C'est une alternative à l'invite de commande de Windows. Il ne fournit pas d'outils supplémentaires mais son utilisation est complémentaire à celle de MinGW. Cette application permet d'exploiter au mieux les ressources fournies par MinGW [23].

3.6.6 Wascana Eclipse

Wascana est la version de la plateforme de développement Eclipse C/C++ comprenant un IDE[14]. Ce logiciel comprend la chaine d'outils de développement MinGW. De plus, la paramètrisation du logiciel permet de travailler directement avec CMake. Un système de plugin permet aussi d'étendre les fonctionnalités du logiciel [24].

Nous avons notamment utilisé plusieurs plugins pour la visualisation des fichiers makefile et pour le développement de l'interface graphique.

[13] Les Makefiles sont des fichiers qui décrivent l'ensemble d'actions (commandes) à exécuter et sur quels fichiers le faire afin de réaliser une tâche.
[14] Environnement de développement intégré.

Chapitre 4 – Application et résultats

Ce chapitre vise à présenter les méthodes de segmentations présentées au chapitre précédent dans le but d'extraire le cartilage de l'articulation du genou. Au terme du mémoire de Sébastien Wilfart [25] plusieurs limitations ont été mises en avant. La première limitation du travail fourni par Sébastien Wilfart est la sous-segmentation du cartilage. En effet, l'application développée par mon prédécesseur ne prend pas les bords du cartilage dans la segmentation, fournissant une évaluation restrictive du cartilage. La seconde limitation est la lourdeur de l'application pour la pose des graines. Dans la suite du chapitre, nous présenterons comment nous avons essayé de répondre à ces deux limites importantes. Les autres points de l'application fournie par Sébastien devant ^tre revus sont surtout un manque d'ergonomie et d'outils complémentaires. Nous présenterons alors les modifications apportées à l'application.

4.1 Démarche de segmentation

L'application présente deux types de segmentations, une dite 2D et l'autre dite 3D. La segmentation 2D correspond à la segmentation d'une coupe après l'autre. Les différents résultats sont alors combinés pour fournir un volume. C'est la méthode que Sébastien a utilisé. Elle présente l'avantage de pouvoir corriger la segmentation en cours de traitement. La segmentation 3D consiste en la segmentation d'un volume. Cette méthode a l'avantage d'être plus rapide mais ne permet pas de corriger le résultat pendant le traitement. Nous allons maintenant présenter les deux méthodes en détail.

4.1.1 La segmentation 2D

L'application développée précédemment utilise la méthode de la croissance de région présentée au chapitre précédent. Le processus de segmentation commence par demander la pose d'au moins une graine pour le départ de la segmentation ainsi que de paramétrer l'intervalle de niveaux de gris. Une fois ces informations données, les images de la série DICOM passent une à une dans l'algorithme de segmentation par croissance de région fourni par ITK. Pour détecter les défauts de segmentation dans une image, le processus compare la zone segmentée de l'image à la zone segmentée de l'image précédente. Ce système utilise le fait qu'il existe peu de différence entre deux images successives de la série. À la fin de la segmentation, la série DICOM des zones segmentées est convertie au format MetaImage. La segmentation du cartilage est alors intégrée dans un seul fichier.

Le problème lié à l'inclusion des bords du cartilage dans la segmentation est dû à l'algorithme de la croissance de région. Pour régler ce problème, l'utilisation d'un autre algorithme est nécessaire. Nous avons tout d'abord essayé les différentes méthodes de segmentation fournies par ITK afin de vérifier que les résultats fournis n'étaient pas meilleurs. Nous sommes arrivés à la conclusion que la méthode de segmentation par la croissance de région choisie par Sébastien donnait les meilleurs résultats. Nous nous sommes donc dirigés vers la coopération de méthodes de segmentation. Des trois méthodes que nous avons présentées, nous avons choisi la coopération de résultats.

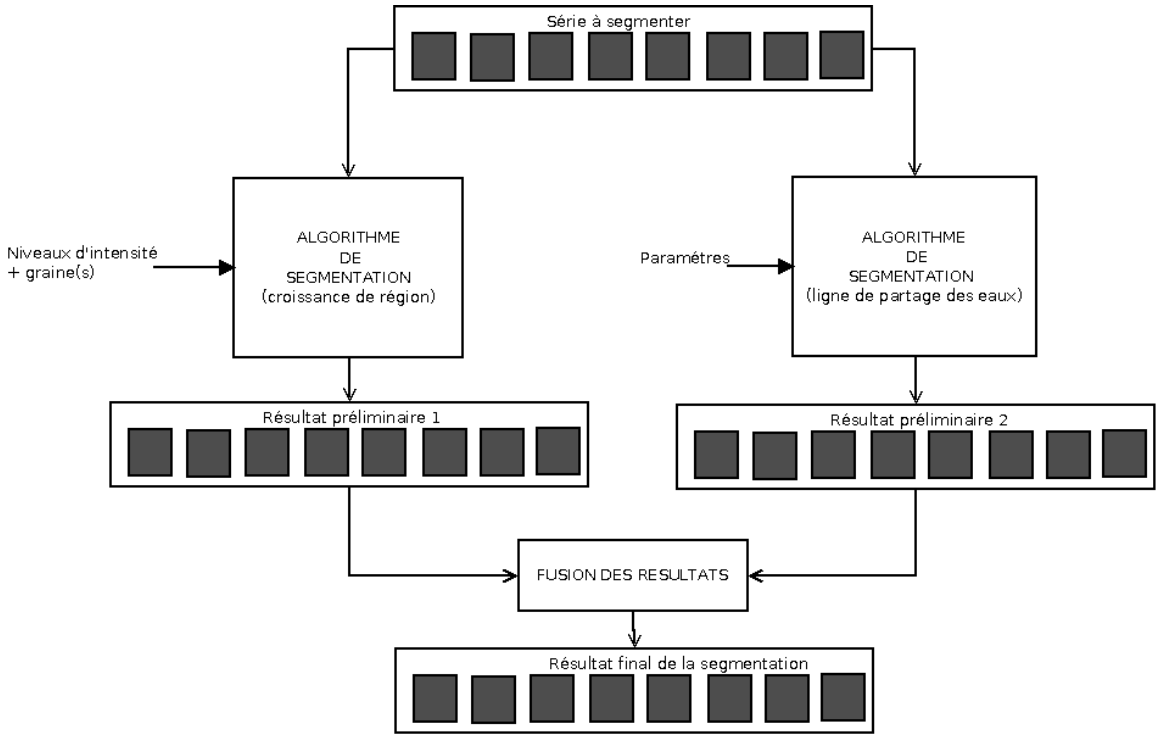


Figure 32 - Méthode de segmentation 2D

Nous avons décidé de garder la méthode initiale et de la compléter à l'aide d'un autre algorithme. Pour que la segmentation ne prenne pas beaucoup plus de temps, l'algorithme de la ligne de partage des eaux a été choisi. En effet, cette technique a l'avantage d'être très rapide mais elle a comme inconvénient de segmenter toute l'image. Pour améliorer la segmentation de l'algorithme de croissance de régions, nous utilisons le résultat de la ligne de partages des eaux de la manière suivante :

- Pour tous les pixels de la zone segmentée par la croissance de région, nous prenons les coordonnées du pixel plus celles du pixel directement adjacent en dehors de la zone segmentée.
- Dans le résultat de la ligne de partage des eaux, nous regardons si les points correspondant aux coordonnées sont dans la même zone de segmentation.
- Si oui, dans le résultat de la croissance de région, nous ajoutons le pixel adjacent dans la segmentation. Si non, nous ne changeons pas la segmentation.

Les pixels ajoutés dans la segmentation de base de la croissance de région ne sont pas pris en compte. L'algorithme de fusion des résultats ne fait donc qu'une seule fois le tour de la segmentation fournie par la méthode de croissance de région.

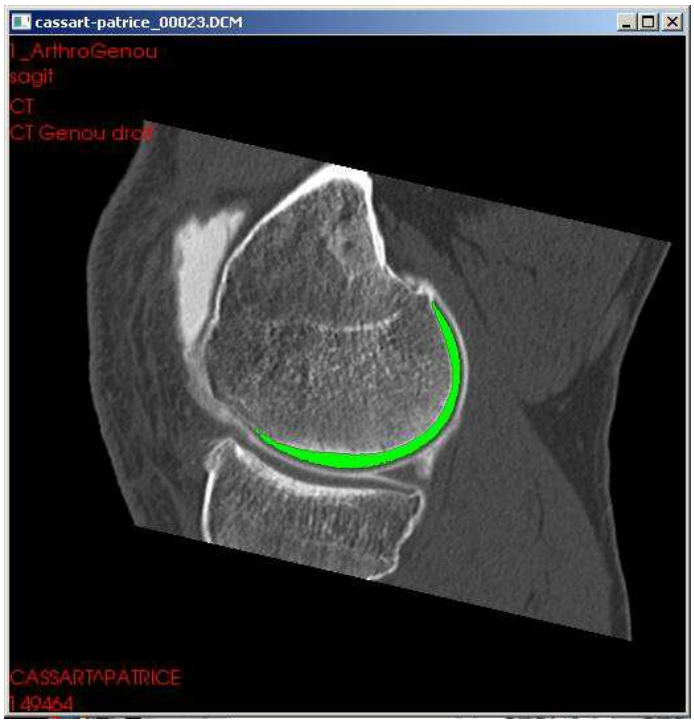


Figure 33 - Segmentation d'une coupe par l'application de Sébastien Wilfart

Figure 34 - La même segmentation traitée par la nouvelle application

Cette technique a permis de mieux épouser le bord du cartilage. Les différentes couleurs présentes sur la segmentation de la nouvelle application donnent une information sur la densité du cartilage. L'échelle de Hounsfield fournit les intervalles de valeurs de niveaux de gris sur une radiographie pour chaque type de matières présentes dans le corps humain. La couleur verte correspond aux pixels se trouvant dans l'intervalle prévu pour le cartilage. La couleur bleu désigne les pixels en dessous de l'intervalle, le cartilage correspondant ayant une trop faible densité. La couleur

rouge signale les pixels qui sont au dessus de l'intervalle prévu, le cartilage correspondant ayant alors une densité trop élevée.

Après correction manuelle des fuites résiduelles, nous obtenons le volume suivant :

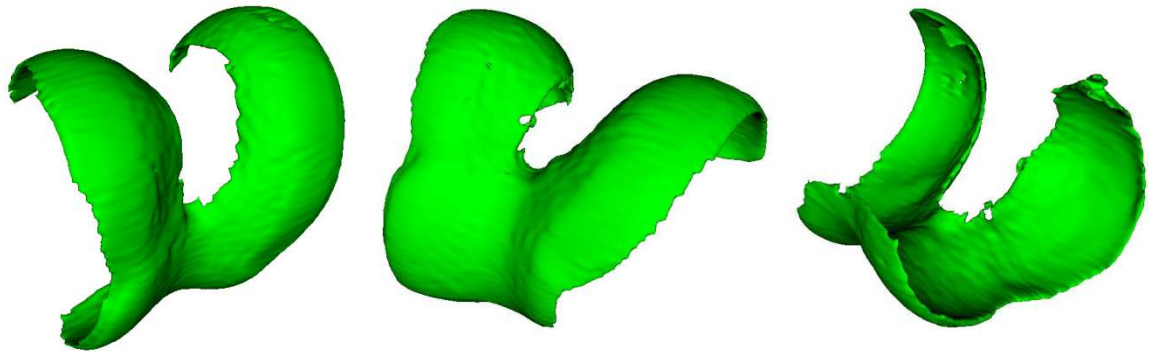


Figure 35 - Volume après segmentation par la méthode 2D

4.1.2 La segmentation 3D

Dans les essais préliminaires, Sébastien Wilfart a voulu utiliser une méthode segmentation en 3D. Bien qu'il ait remarqué que ces méthodes sont plus rapides que les méthodes 2D, il n'a pas réussi à l'implémenter suite aux trop grandes corrections à apporter. En effet, le phénomène des fuites qui peuvent être corrigées coupe par coupe dans la méthode 2D est accentué dans les méthodes 3D. Afin d'apporter les corrections nécessaires, il faut un modèle de référence qui permette d'évaluer ce que devrait être la zone segmentée. Ce modèle de référence est apporté par les méthodes guidées par atlas.

Comme énoncé précédemment, un atlas est un résultat préalablement obtenu. Nous avons donc dû construire notre atlas avant de pouvoir implémenter une méthode de guidage par atlas. Pour ce faire, nous avons utilisé la méthode de segmentation 2D détaillée à la section précédente pour obtenir une segmentation de qualité. Cette étape a été répétée sur plusieurs genou afin d'avoir un atlas suffisant (cinq résultats de genoux sains et quatre résultats de genoux abimés). Après quelques recherches, la décision fut prise de commencer par un guidage par atlas mono résultat. L'atlas choisi pour correspondre au mieux au genou que nous voulions segmenter devait subir une déformation. Pour effectuer cette transformation de l'atlas, nous avons utilisé l'algorithme « Diffeomorphic Demons ». Une fois la déformation effectuée, nous utilisons l'atlas déformé pour déterminer les graines de départ pour l'algorithme de segmentation par croissance de région. Pour que les graines soient placées sur la bonne coupe, la série originelle de l'atlas doit subir un recalage ou redécoupe afin d'obtenir le même nombre de coupes.

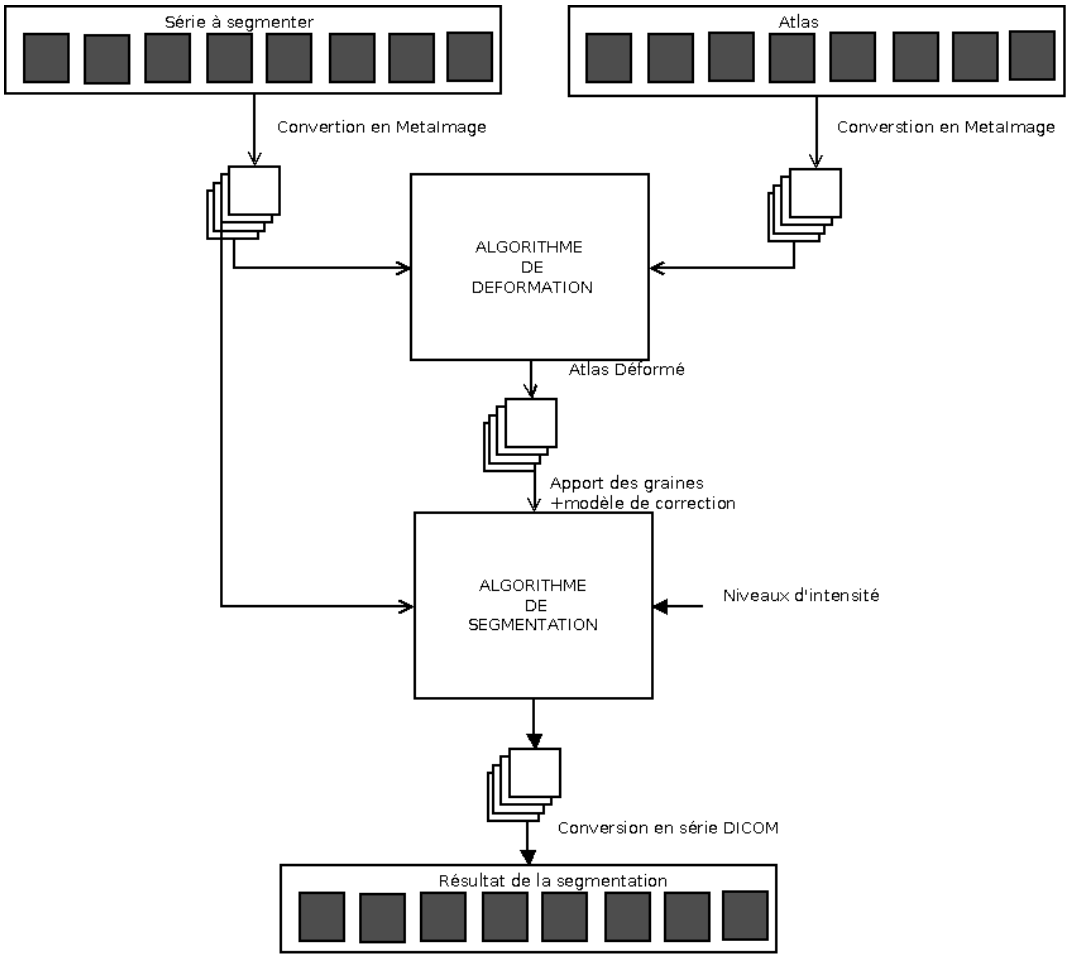


Figure 36 - Méthode de segmentation 3D

Lorsque la segmentation est terminée, nous rencontrons les mêmes problèmes que Sébastien. En effet les fuites présentes sur le résultat brut englobent presque l'intégralité des tissus mous. C'est avec l'atlas que nous allons pouvoir résoudre ce problème qui avait empêché Sébastien de continuer dans cette voie. Vu que l'atlas a été déformé pour correspondre au mieux au genou que nous segmentons, nous utilisons l'atlas pour effacer les fuites présentes. Toutefois, ce procédé n'est pas parfait car l'atlas en lui-même n'est pas l'exacte réplique du genou que nous voulons segmenter.

Dans la procédure de segmentation 3D, nous n'avons pas su utiliser tous les algorithmes de la procédure 2D. Ce changement de technique s'explique par une utilisation trop importante de mémoire dans les procédures 3D. Pour une utilisation sans risque de manque de mémoire des algorithmes 3D, il est conseillé par la communauté ITK de disposer de 8 à 16 gigaoctets de mémoire RAM. Nous avons également rencontré des problèmes similaires dans la procédure de déformation. Dans ce cas précis nous avons effectué un « Downsampling » des images pour déformer l'atlas sur qui nous avons ensuite appliqué un « Upsampling » afin de revenir aux dimensions de base.

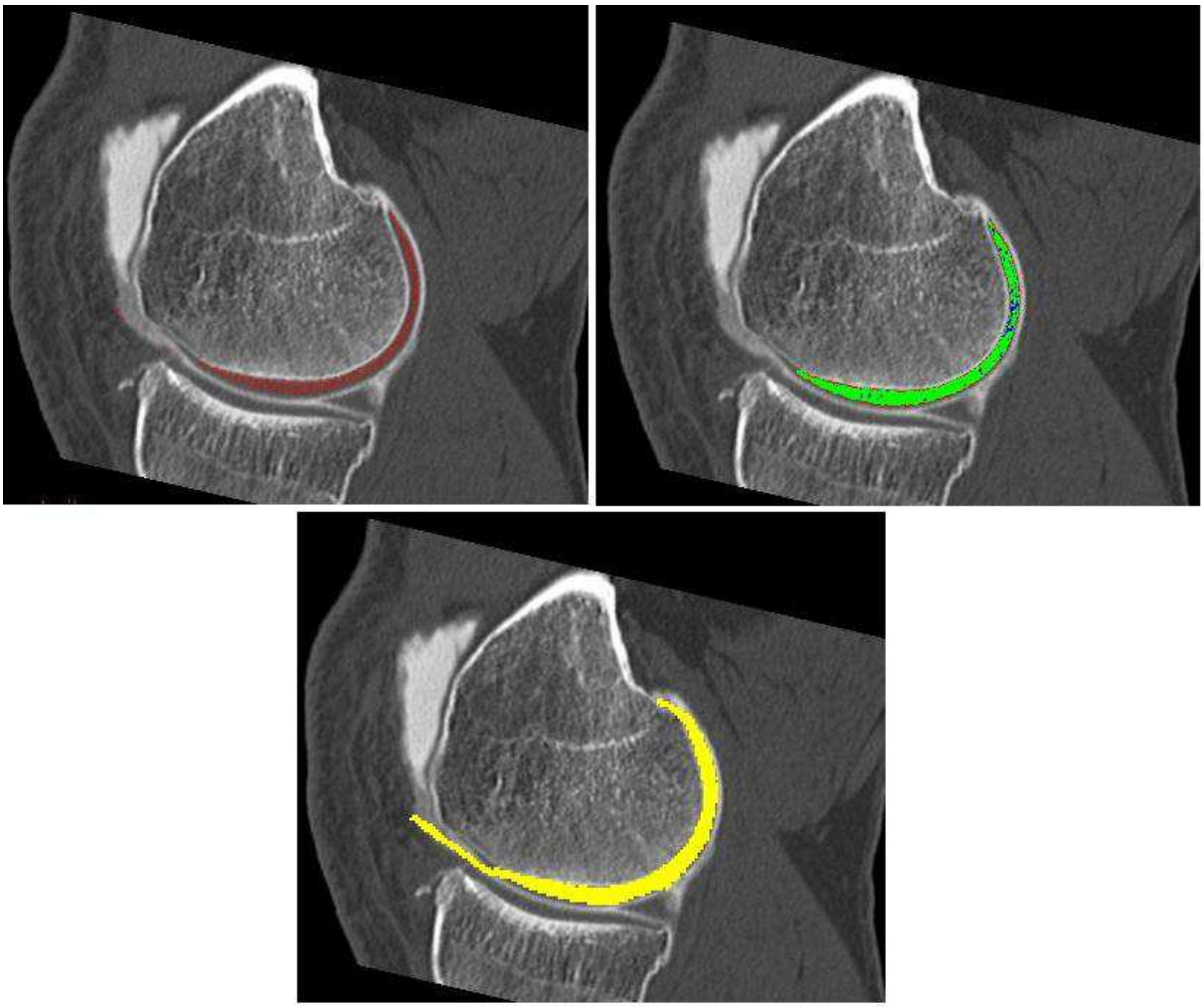


Figure 37 – En haut à gauche, l'image comportant les graines pour la coupe. En haut à droite, le résultat final pour la coupe. En bas, l'atlas déformé pour la coupe

Sur la figure ci-dessus, nous pouvons voir un exemple de résultat pour une coupe. L'atlas déformé est affiché en jaune sur la coupe, on peut voir que la déformation effectuée n'est pas parfaite. De cet atlas, nous tirons les graines de base ; elles sont affichées en rouge. Après correction nous obtenons le résultat affiché vert principalement. On remarque aisément que la zone segmentée présente le même défaut que dans l'application de Sébastien, c'est-à-dire que la zone segmentée n'épouse pas complètement le bord du cartilage. En effet, vu que nous n'utilisons que l'algorithme de croissance de région, nous constatons les mêmes limitations.

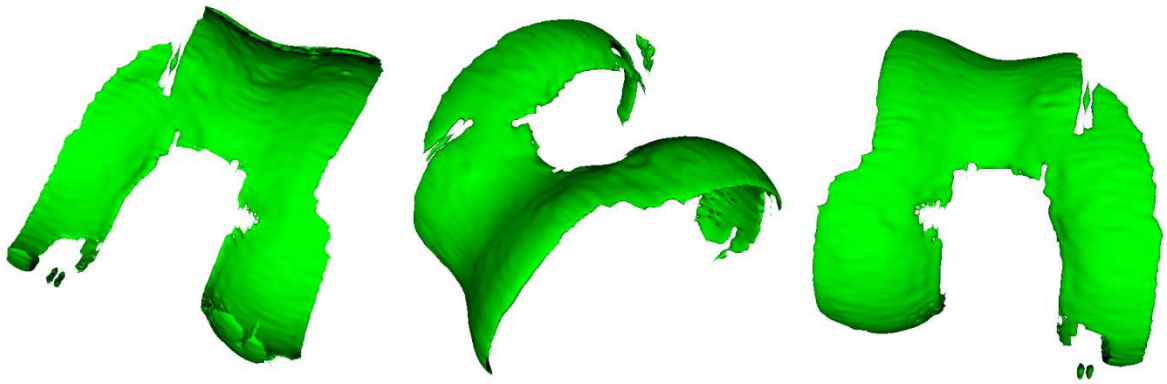


Figure 38 - Volume après segmentation par la méthode 3D

Le volume repris ci-dessus est le résultat de la segmentation par le processus 3D sans aucune correction. On remarque la présence de trous dans le volume, dus à l'inégalité de l'atlas par rapport au genou à segmenter. Malgré ce désavantage, cette méthode présente beaucoup moins de fuites résiduelles.

De plus, le choix de l'atlas est très important pour avoir les meilleurs résultats possibles. La méthode guidée par atlas de notre application est encore très peu évoluée. En effet, l'application n'utilise qu'un seul atlas, donc si cet atlas n'est pas ressemblant au genou à segmenter, les résultats seront médiocres. Dans ce cas, il faut que l'utilisateur effectue le changement d'atlas manuellement dans le système de fichier.

4.2 Le GUI

Durant son stage, Sébastien a dû se concentrer sur le cœur du projet qui est la segmentation du cartilage du genou. Ne partant de rien, il n'a pas eu la possibilité d'améliorer l'interface graphique. Son interface est donc principalement constituée de lignes de commande et de fenêtres de visualisation de VTK (une pour la visualisation des coupes, une pour le volume et une troisième pour l'outil visualisation avec superposition multi-planaire), comme le montre la figure ci-dessous. Cette interface demande donc de jongler avec plusieurs fenêtres pour effectuer une action. En effet, pour lancer un outil, il faut connaître la touche correspondante que l'on peut retrouver dans la fenêtre en ligne de commande, de mettre le focus[15] sur la fenêtre sur laquelle on veut exécuter l'outil et de presser la touche. Si cet outil demande en plus des arguments, il faut remettre le focus sur la fenêtre de ligne de commande pour introduire les paramètres.

Par exemple, pour paramétrer l'algorithme de segmentation, le focus doit être maintenu sur la fenêtre de visualisation des coupes et la touche « p » doit être enfoncée. Une fois la touche pressée, la fenêtre en ligne de commande change et le focus doit être remis dessus pour introduire les arguments. Voyons ce que cela donne à l'aide d'images.

[15] Une fenêtre sur laquelle il y a le focus est la fenêtre avec laquelle on interagit. Sur Windows, c'est celle dont la barre de titre est bleue.

Nous avons pour le moment le focus sur la fenêtre de visualisation des coupes.

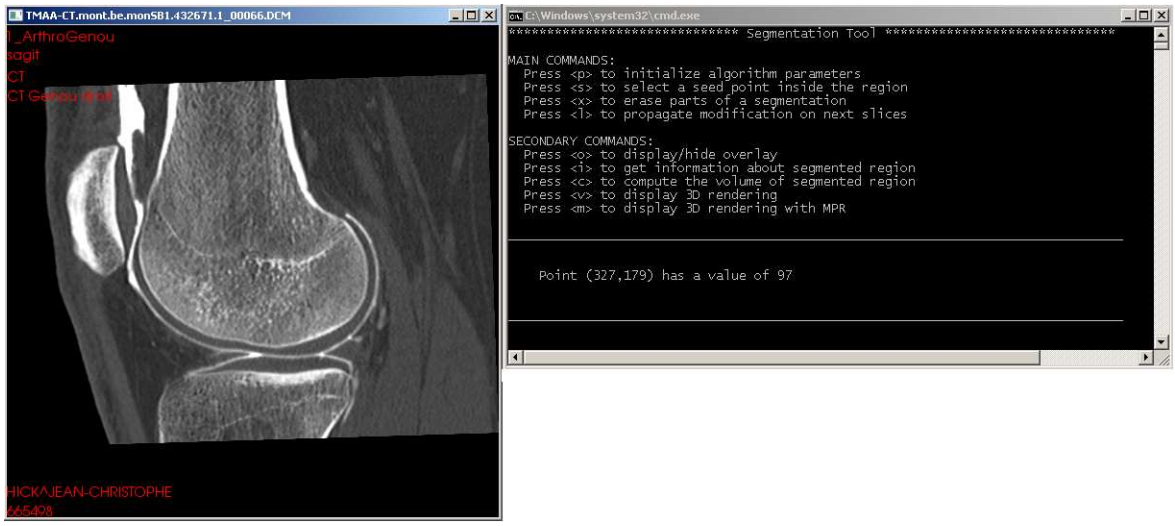


Figure 39 - Interface de l'application de Sébastien Wilfart

Après avoir pressé la touche « p » et passage du focus sur la fenêtre de ligne de commande, nous obtenons :

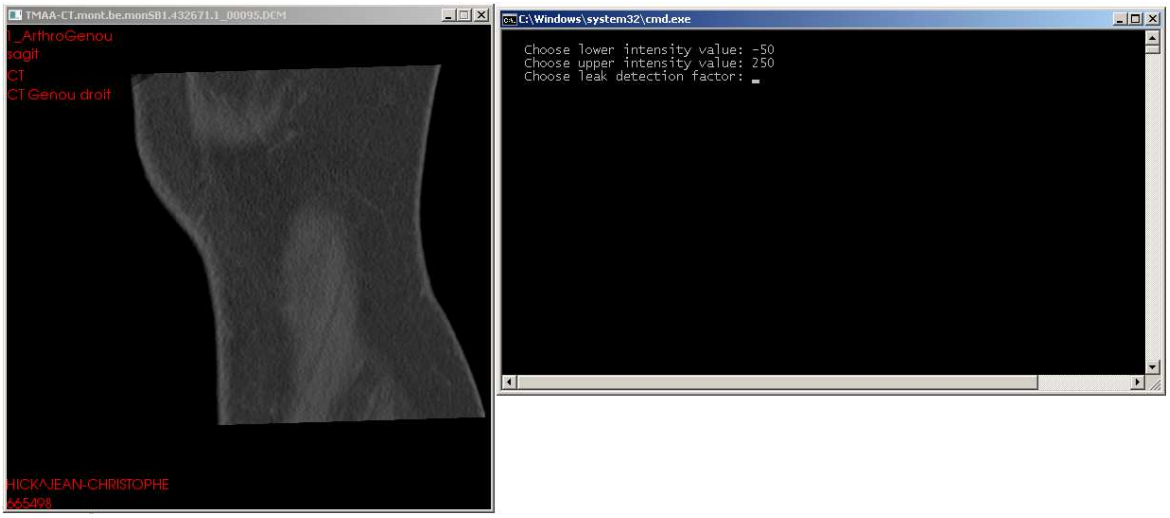


Figure 40 - Préparation de l'algorithme de segmentation

Une fois les paramètres introduits dans l'application, nous devons remettre le focus sur la fenêtre de visualisation des coupes pour lancer un autre outil tel que la pose de graines.

Dans un souci de facilité pour l'utilisateur, nous avons essayé de regrouper toutes ces fenêtres en une seule. Cela n'a pas été possible. Toutefois, nous avons réussi à créer une fenêtre principale dans laquelle les autres fenêtres s'intègrent. De plus, la fenêtre en ligne de commande bien qu'encore apparente, n'est plus utilisée. L'utilisateur ne doit plus qu'interagir avec la fenêtre principale et les fenêtres pop-up pour l'introduction de paramètres. Nous avons aussi opté pour une utilisation maximale de la souris en créant des boutons d'activation des différents outils.

Le résultat final est présenté par la figure suivante. Il faut tout de même mentionner que lorsque plusieurs fenêtres sont ouvertes dans la fenêtre principale, le focus ne se fait que sur une

seule d'entre elles. Ainsi, lorsque le focus est sur la fenêtre de visualisation des coupes, il faut cliquer sur la fenêtre de visualisation du volume avant de pouvoir effectuer une rotation.

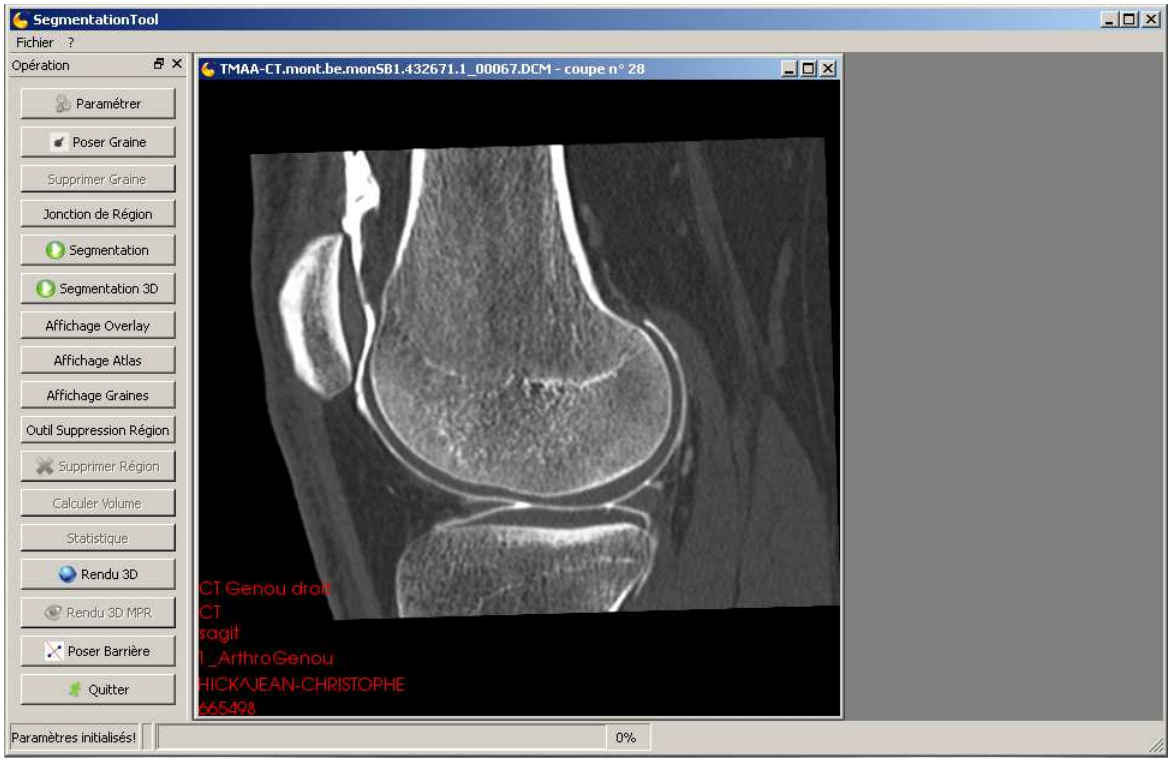


Figure 41 - Nouvelle interface implémentée avec Qt

Un exemple de fenêtre pop-up est la fenêtre des paramètres des algorithmes. Cette fenêtre apparait lorsqu'on appuie sur le bouton « paramétrer ». Les différents champs de paramètres sont déjà remplis avec une configuration de base qu'il faudra adapter au cas par cas.

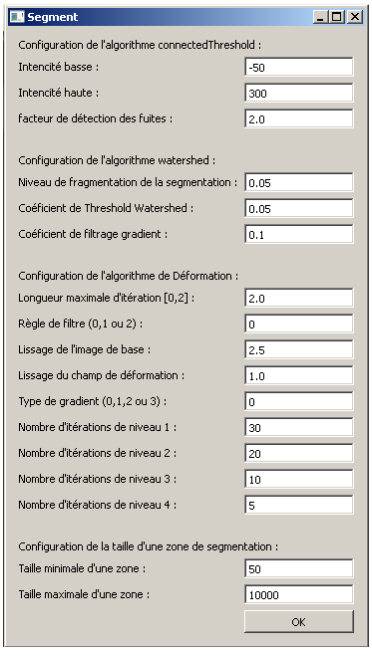


Figure 42 - Fenêtre des paramètres des algorithmes

Pour quitter l'application proprement, il faut impérativement utiliser le bouton « Quitter ». L'utilisation de la croix (en haut à droite d'une fenêtre) ne supprime pas proprement les objets. Cela entraine un arrêt critique de l'application. Le même phénomène est constaté avec la fenêtre de visualisation 3D. Dans ce cas, il faut appuyer sur le bouton « rendu 3D » pour lancer l'outil et réappuyer sur le bouton, qui aura changé d'intituler et portera l'inscription « Quitter Rend 3D » pour le couper.

4.3 Les outils complémentaires

Nous présenterons maintenant les outils qui ont été ajouté à l'application afin de faciliter légèrement l'utilisation. Ces outils touchent principalement à la pose de graines ainsi qu'à la visualisation des coupes

4.3.1 Pose de graines

L'outil de pose de graines fourni avec l'application de Sébastien demandait de presser la touche « s » pour poser une graine. Si l'utilisateur voulait en poser une deuxième, il devait appuyer une nouvelle fois sur la touche en question et ainsi de suite.

Lorsque nous sommes passés à une interface graphique avec bouton, l'application souffrait encore de cette limitation. Afin de pallier à ce problème, il a été décidé de créer un deuxième mode de pose de graines.

- Dans le premier mode, l'utilisateur clique une fois sur le bouton « Poser Graine » et clique ensuite sur la coupe à l'endroit où il veut placer la graine. Donc pour un clic correspond une graine.
- Dans le deuxième mode, l'utilisateur clique deux fois sur le bouton « Poser Graine ». Le bouton devient alors bleu. À ce moment, l'utilisateur peut poser autant de graine qu'il le souhaite. Pour mettre fin à ce mode, l'utilisateur clique une fois encore sur le bouton. Ce dernier reprend alors sa couleur d'origine.

Figure 43 - Bouton de pose de graines dans la pose multiple de graines

4.3.2 Suppression de graines

Il a été remarqué que lorsque l'utilisateur posait sa graine au mauvais endroit, cela pouvait engendrer la prise en compte d'une région non voulue. Pour corriger cette erreur, l'utilisateur devait utiliser l'outil de suppression de région pour enlever la zone non voulue. Bien que cet outil soit très utile pour supprimer les fuites résiduelles, il n'est pas très performant dans ce cas.

Un outil permettant de supprimer la dernière graine posée a donc été développé afin de faciliter la correction. Pour ce faire, l'utilisateur doit simplement cliquer sur le bouton « Supprimer Graine ».

4.3.3 Zoom

Lorsque le résultat d'une segmentation présente des défauts, il est nécessaire d'ajouter des graines ou de supprimer des parties de la segmentation. Ces corrections demandent une plus grande

précision que les opérations de bases. La taille des images n'étant pas très élevée, la précision requise pour mener à bien les corrections nécessaires dépend de l'attention et de la dextérité de l'utilisateur.

Pour que ces corrections soient plus aisées à opérer et que l'utilisateur puisse mieux observer le résultat, un outil de zoom sur l'image a été ajouté à l'application. Le zoom s'applique à la fois sur l'image médicale et sur les différents affichages qui se superpose à l'image médicale. Pour appliquer un zoom, l'utilisateur doit maintenir le bouton droit de la souris enfoncé et effectuer un mouvement vers l'avant ou l'arrière.

4.4 Problèmes généraux

Au terme du stage de Sébastien, certains bugs étaient restés présent dans l'application. Notre première tâche a été de les corriger. Le plus important était d'utiliser une taille d'image fixe. Bien que toutes les images fournies par le logiciel Telemis soient de dimension 512 pixels sur 512 pixels, il est préférable de vérifier la taille de l'image afin de ne pas dépasser les limites de l'image. Cette vérification permet aussi une plus grande évolutivité de l'application.

Nous pouvons aussi mentionner la présence d'un léger décalage vers le haut de l'affichage de la segmentation. Ce problème a été réglé par le passage de l'interface vers Qt qui demandait l'utilisation d'objets particuliers.

4.5 Résultats

Dans tout le processus de développement, nous nous sommes axés sur la segmentation du cartilage fémoral en coupes sagittales. Toutes les données fournies dans ce qui suit concernent donc la segmentation du cartilage fémoral par les coupes sagittales. Pour obtenir des résultats cohérents, nous avons utilisé la même configuration pour l'algorithme de segmentation par croissance de région dans les deux applications. Cette configuration varie entre les différents patients.

Les résultats sont en deux parties, la première concerne les segmentations complètes fournies par les différents algorithmes (c.à.d. sans correction manuelle). La deuxième partie concerne les segmentations après suppression des fuites. Cela donne une idée de la performance des différentes techniques.

Tableaux des volumes sans correction :

Patient	Segmentation Sébastien	Segmentation 2D	Segmentation 3D
Patient 1	39412 mm³	11640 mm³	7527 mm³
Patient 2	98896 mm³	16339 mm³	14505 mm³
Patient 3	53942 mm³	20704 mm³	8928 mm³
Patient 4	26534 mm³	15058 mm³	8608 mm³
Patient 5	11916 mm³	11526 mm³	3818 mm³

Tableaux des volumes après suppression des zones excédantes :

Patient	Segmentation Sébastien	Segmentation 2D	Segmentation 3D
Patient 1	10475 mm³	11405 mm³	6527 mm³
Patient 2	12003 mm³	12714 mm³	9789 mm³
Patient 3	7244 mm³	8683 mm³	8406 mm³
Patient 4	7955 mm³	10095 mm³	6118 mm³
Patient 5	4075 mm³	5708 mm³	3492 mm³

À ce stade, nous remarquons que la méthode 3D est loin d'être au point. Bien que prometteuse, la technique de construction ou de choix de l'atlas doit être revue pour fournir des résultats corrects. Nous pouvons remarquer, dans le cas du patient 3, que la méthode de l'atlas est très performante lorsque l'atlas utilisé ressemble au genou à segmenter.

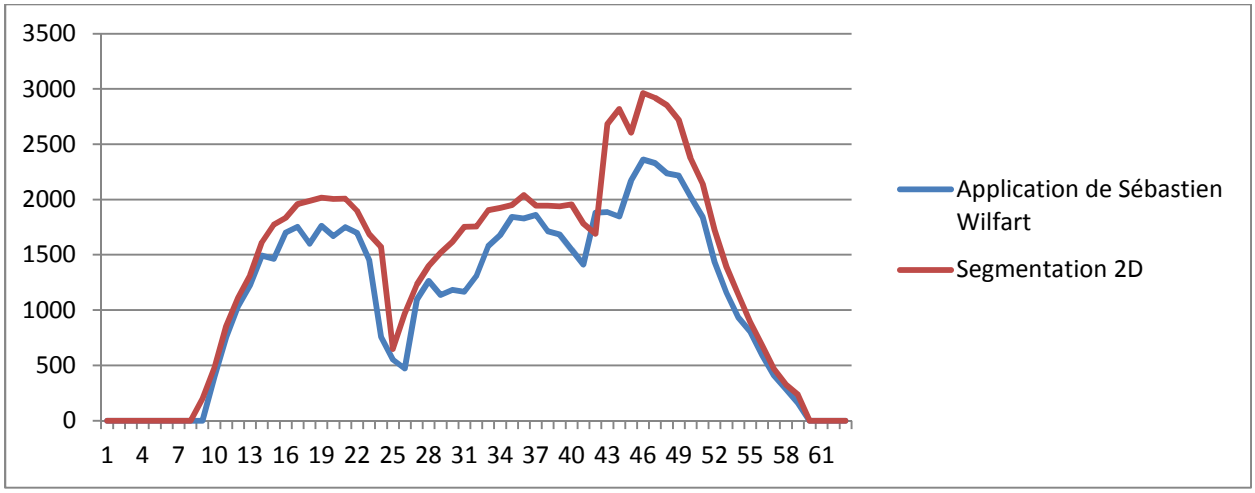
Concernant les deux techniques 2D, les résultats de la nouvelle technique comportent, de manière générale, moins de fuites et nécessitent donc moins de correction de suppression. Pour aller plus loin dans la comparaison de ces deux techniques, nous allons fournir les volumes des segmentations des trois derniers patients après l'ajout des parties du cartilage qui n'ont pas été reprises dans la segmentation.

Tableaux des volumes après ajout des zones manquantes :

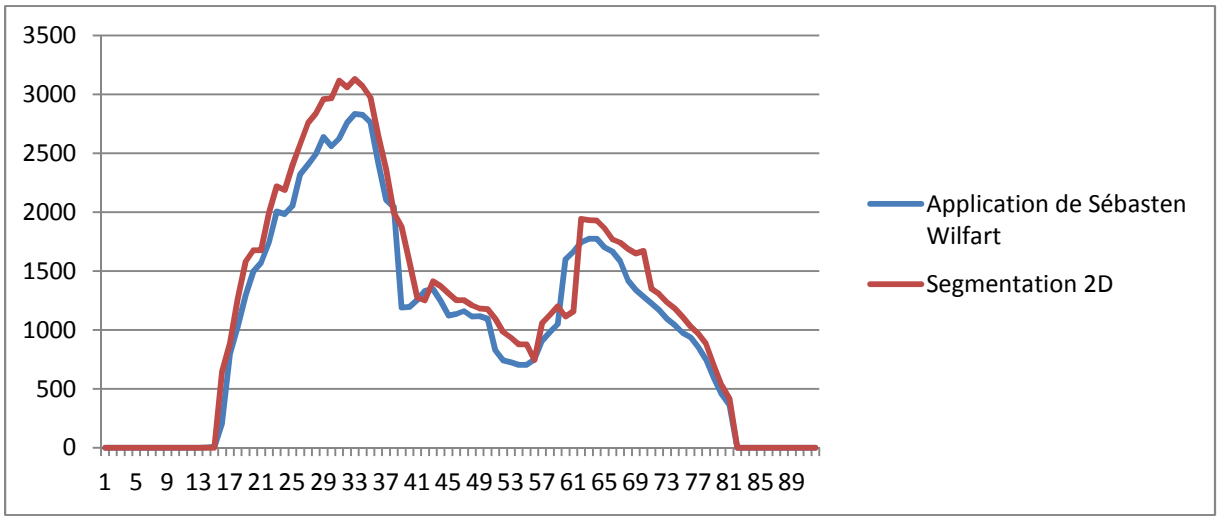
Patient	Segmentation Sébastien	Segmentation 2D
Patient 3	7504 mm³	8718 mm³
Patient 4	9068 mm³	10192 mm³
Patient 5	5297 mm³	6260 mm³

Nous remarquons que les différences de volume de segmentation entre les deux techniques sont plus petites après avoir ajouté les zones de cartilage non segmentée. Toutefois, nous remarquons que les différences de volumes restent conséquentes. La nouvelle technique de segmentation en 2D réussi donc à se rapprocher plus des bords du cartilage, incluant ainsi plus volume.Pour vérifier ces observations, nous allons donner les graphiques des trois derniers patients donnant le nombre de pixels pris dans la segmentation par coupe.

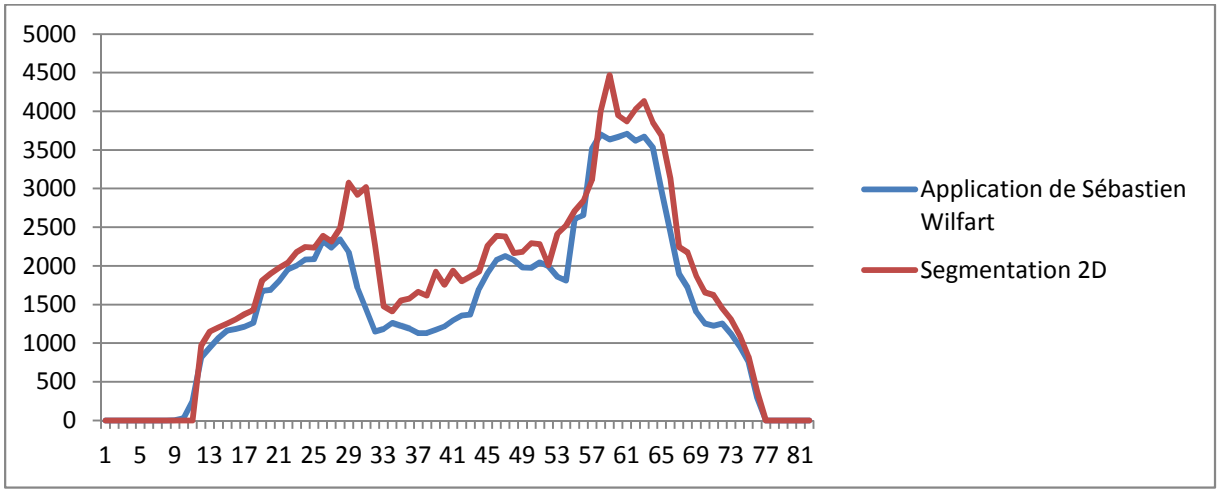
Graphe pour le patient 3 :



Graphe pour le patient 4 :



Graphe pour le patient 5 :

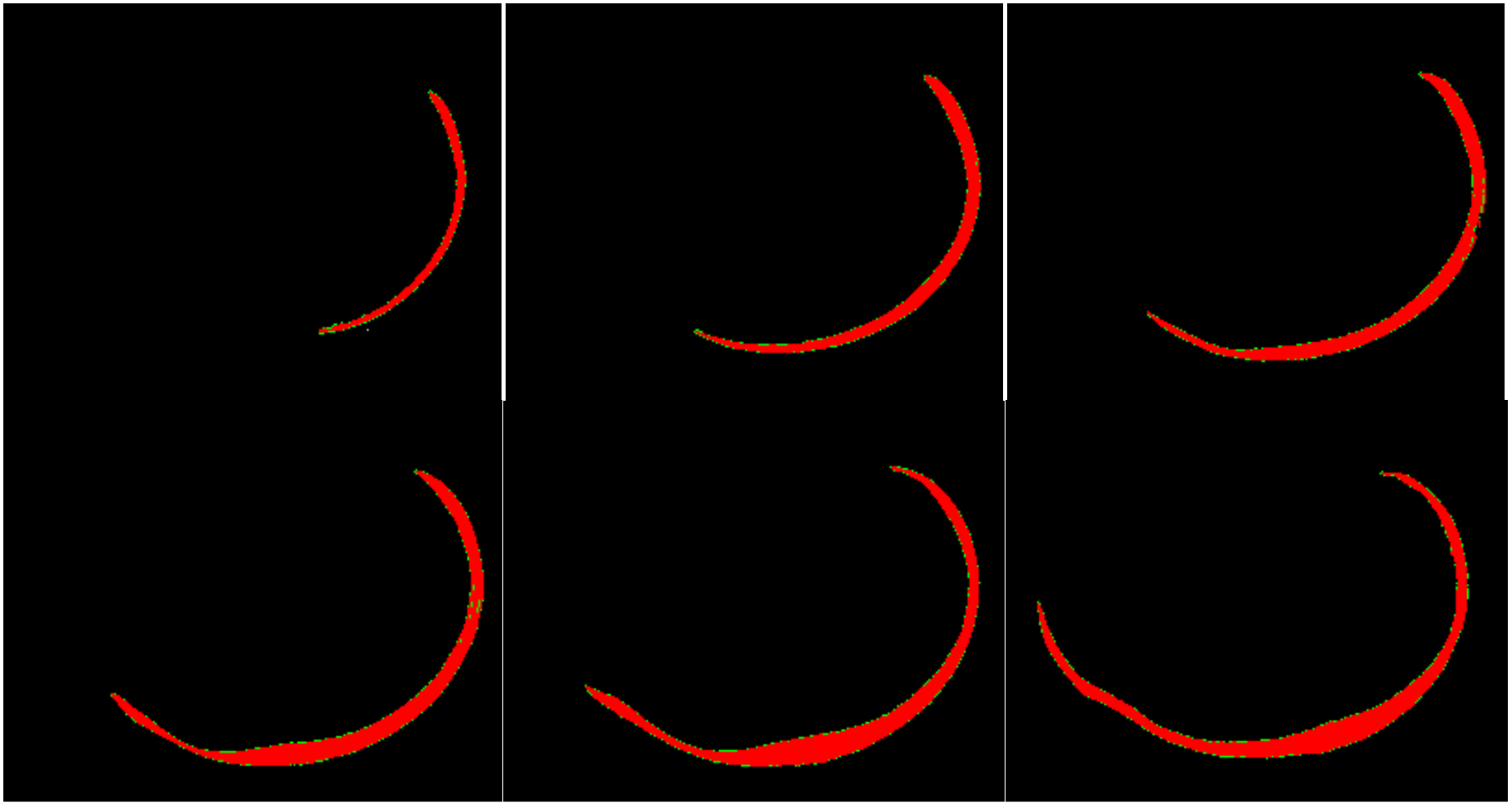


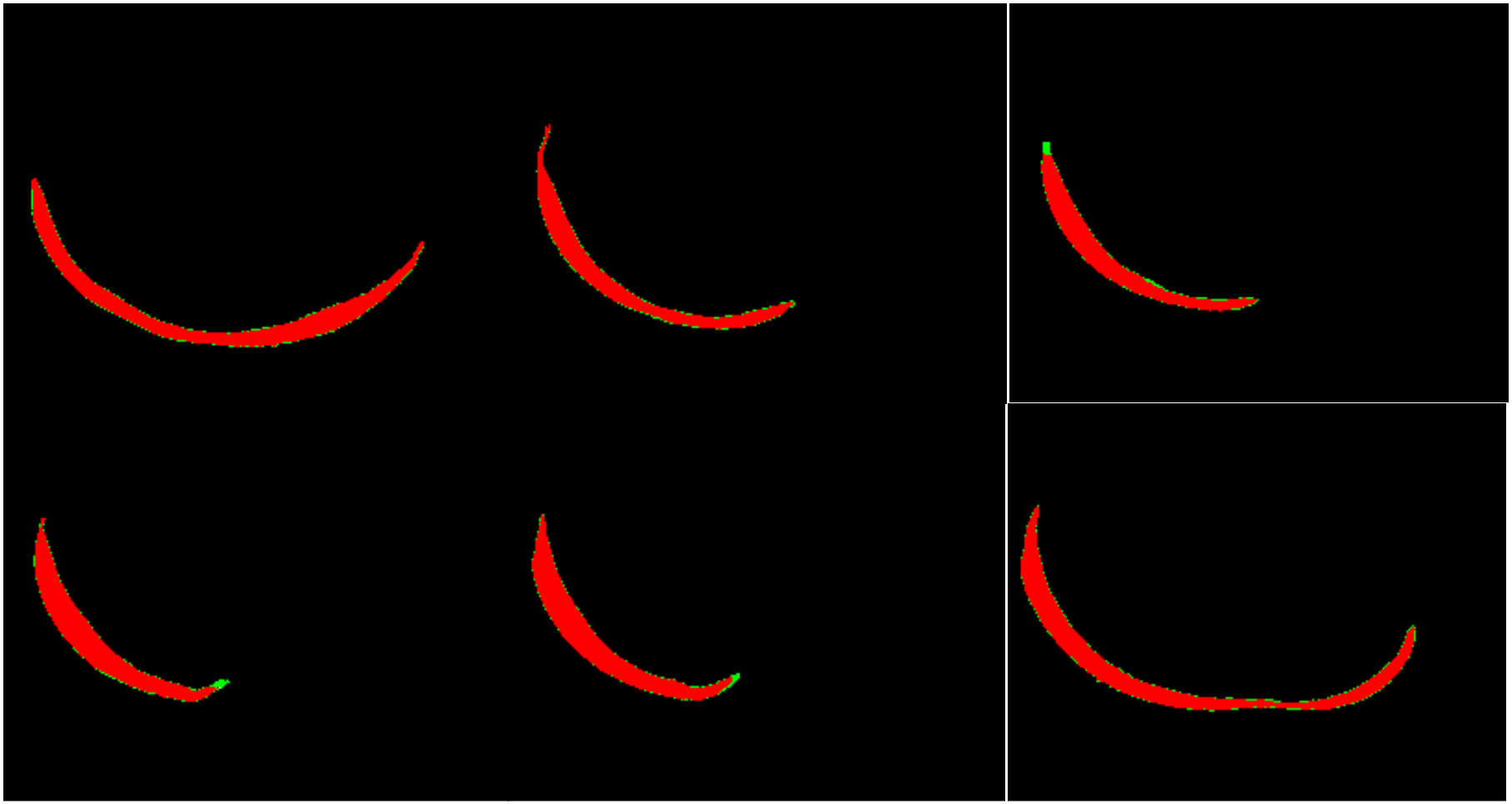
Après observation des graphiques, nous constatons que la nouvelle méthode de segmentation englobe un peu plus de pixels sur la plupart des coupes. Pour constater cette différence, nous avons superposé la segmentation de différentes coupes obtenue par l'application de Sébastien sur la segmentation de la nouvelle méthode 2D. Nous avons opté pour cette solution car la visualisation côte à côte des résultats n'est pas aisée (voir les deux figures sur la page suivante).

Figure 44 - À gauche, la segmentation de Sébastien Wilfart; À droite, la segmentation de la méthode 2D

Figure 45 - À gauche, la segmentation de Sébastien Wilfart; À droite, la segmentation de la méthode 2D

Vu que la nouvelle méthode de segmentation 2D tend à prendre plus de pixel, nous avons positionné cette dernière en fond que nous avons coloré en vert. Sur ce résultat nous avons superposé le résultat de la méthode de Sébastien que nous avons coloré en rouge. Ainsi, nous pouvons plus facilement observer les pixels que la nouvelle méthode ajoute à la zone d'intérêt. Les résultats ci-dessous ne représentent qu'un échantillon des coupes du patient 1. Les coupes ont été prises à un intervalle régulier de toutes les quatre coupes afin de se donner une idée générale des différences dans les deux résultats.





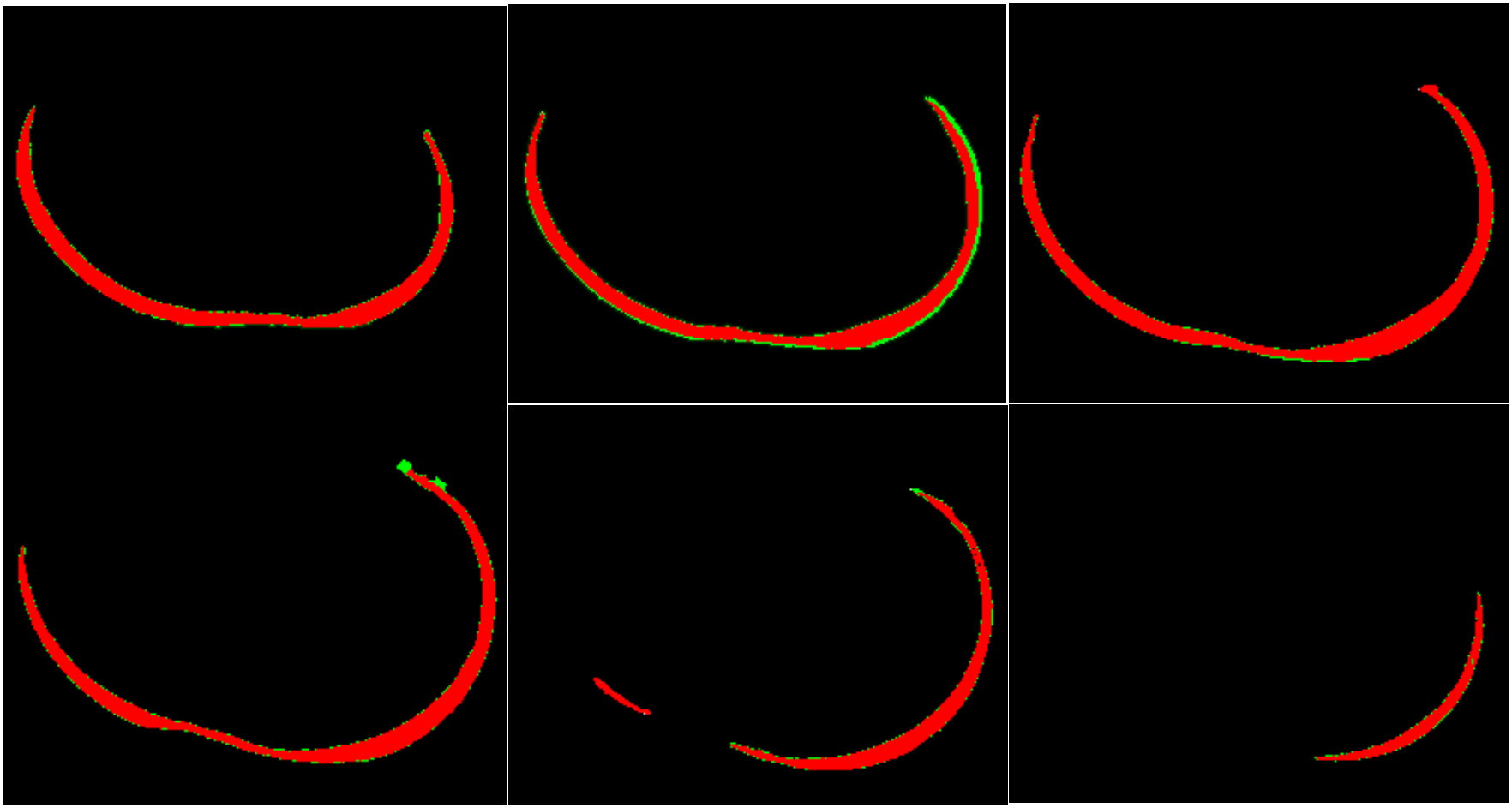


Figure 46 - Comparaison de résultat méthode 2D; En rouge la segmentation par l'application de Sébastien, en vert, les ajouts de la nouvelle méthode

Afin de constater les manquements de la méthode 3D nous avons procéder à la même comparaison entre la nouvelle méthode 2D et la méthode 3D. Nous avons mis le résultat de la méthode 2D coloré en vert en fond et le résultat de la méthode 3D en rouge. Vu le travail qui est encore à fournir afin que la méthode 3D soit satisfaisante, nous nous sommes limité à une comparaison sur 6 coupes du patient 1.

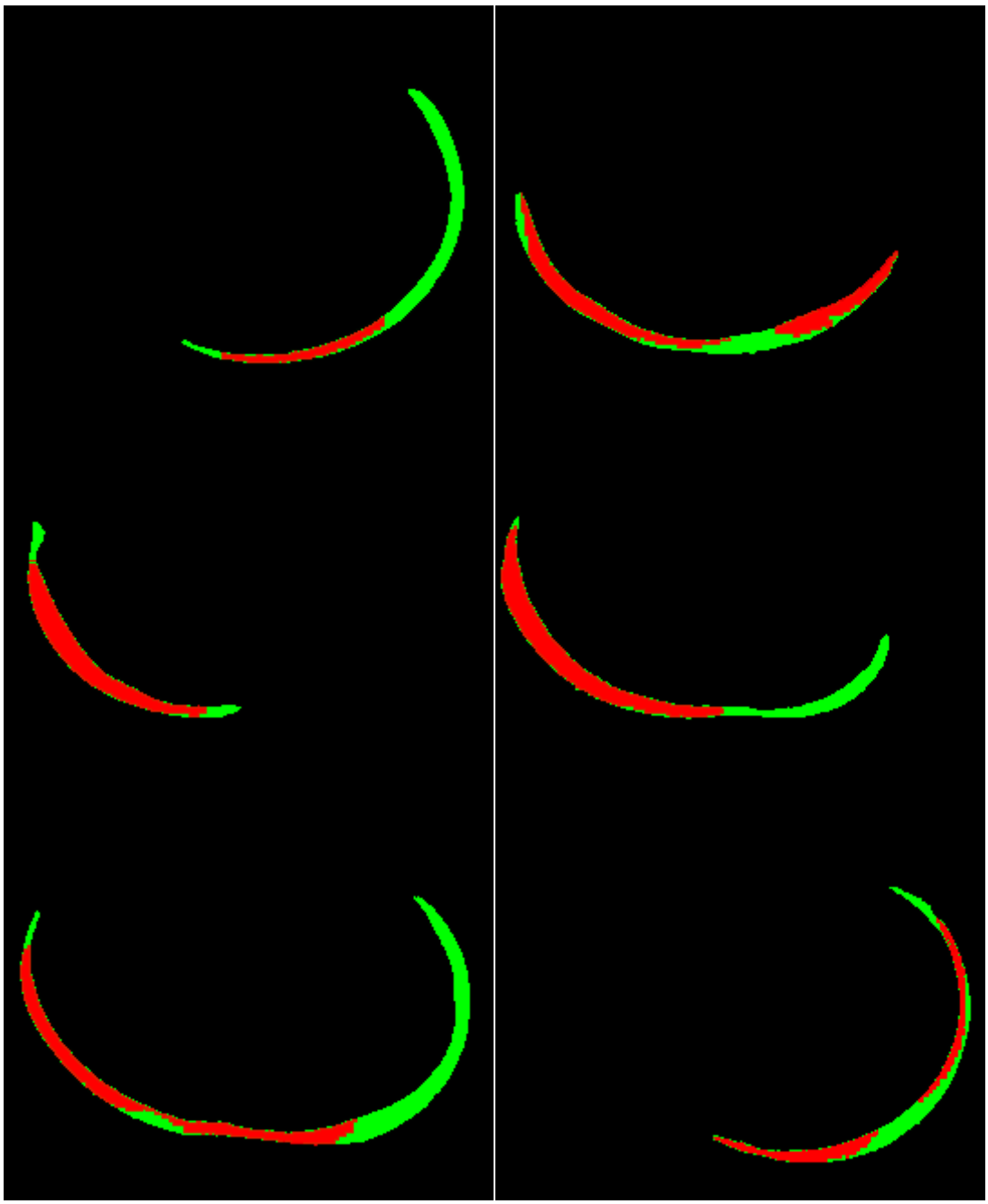


Figure 47 - Comparaison de résultat méthode 3D; En rouge la segmentation par la méthode 3D, en vert, celle de la méthode 2D

Nous pouvons remarquer qu'une grande partie du cartilage n'a pas été pris dans la segmentation. Cela est du à l'utilisation d'un atlas fort différent que l'algorithme de déformation n'arrive pas à faire correspondre au genou à segmenter. L'amélioration de la méthode de choix de l'atlas est donc primordiale.

4.5.1 Temps de segmentation

Si la qualité des résultats finaux a été améliorée, le temps d'une segmentation a, au contraire, été détérioré. L'ensemble des traitements automatiques s'ajoutant aux algorithmes de segmentations allongent le temps que prends tout le processus. À cela, il faut encore rajouter le temps passé pour les corrections manuelles. Durant le développement de l'outil, le temps de l'exécution n'a pas été un facteur décisif. Toutefois, il est intéressant de connaître quelques mesures sur le temps d'exécution et quelles sont les étapes qui prennent le plus de temps.

Pour la segmentation en 2D, une segmentation sans correction manuelle prend entre 15 à 20 minutes. Pendant ce temps l'utilisateur ne peut rien faire car l'application fonctionne sur un seul thread. Ce long laps de temps s'explique par l'exécution de deux algorithmes de segmentation suivis d'une méthode de correction de fuites et de zones inintéressantes, d'une méthode de mise en commun des résultats. Enfin vient l'ajout de l'information sur la densité du cartilage.

Ce sont les méthodes de correction, de mise en commun et d'ajout d'information qui allongent le temps de façon significative. Les algorithmes en eux-mêmes ne demandent pas beaucoup de temps d'exécution.

Pour la segmentation par la méthode 3D, le temps d'une segmentation sans correction manuelle peut dépasser la demi-heure. Cela s'explique par l'algorithme de déformation, utilisé pour faire correspondre l'atlas et le genou à segmenter, qui prend à lui tout seul entre 15 et 20 minutes. Ensuite vient l'exécution de l'algorithme de segmentation, les corrections de fuites et suppression de zones inintéressantes.

Dans la segmentation en 3D les étapes de transformation d'une série DICOM en volume et inversement, demande environ 5 minutes à elles deux. Cela ne reste qu'une dizaine de minute maximum pour la segmentation et sa correction.

Chapitre 5 – Conception

Ce chapitre fournit les informations essentielles pour toutes personnes désireuses de poursuivre le projet. En effet, ce chapitre reprend l'architecture du projet, explique les liens entre les classes et le fonctionnement de l'outil.

5.1 Architecture

Dans la lignée de Sébastien Wilfart [25], nous avons essayé d'utiliser les librairies ITK et VTK sous l'angle d'une approche problème et non traitement. Ainsi, chaque classe du projet répond à un problème donné. Plusieurs classes peuvent utiliser les mêmes traitements offerts par les librairies ITK et VTK mais leurs utilisations diffèrent en fonction de l'objectif fixé.

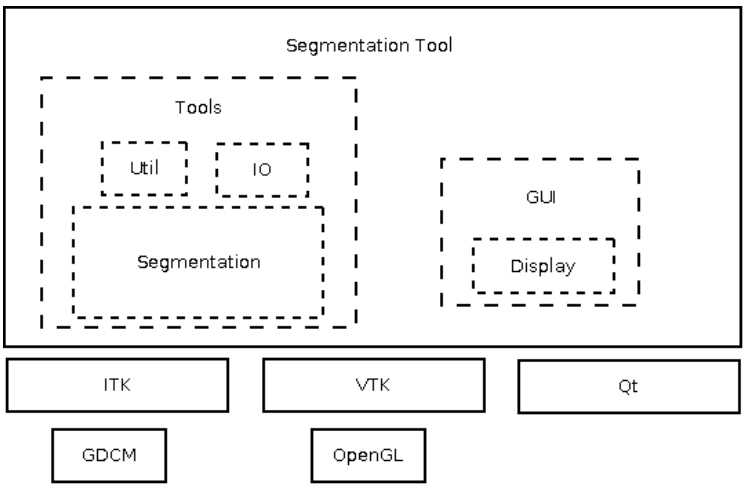


Figure 48 - Architecture globale du projet

Le projet est donc divisé en deux grandes parties : l'interface et les outils. Ces deux parties sont elles-mêmes composées d'autres parties. La partie « GUI » est composée de l'interface graphique développée sur les librairies Qt et d'une sous-partie « Display » qui utilise les outils de visualisations fournis par VTK. La partie « Tools » reprend tous les outils répondant à un problème. Ces outils sont divisés en fonction du type de problème. Les outils de la partie « IO » répondent aux problèmes de lecture et d'écriture de fichiers sur le disque dur. La partie « Segmentation » offre les outils de segmentation d'image et la partie « Util » fournit les outils qui englobent la segmentation.

Bien que Sébastien ait utilisé cette architecture pour créer des package dans le projet, l'utilisation de Eclipse en composition de Cmake rend l'utilisation de package très lourde. Nous avons donc gardé cette décomposition théorique dans tout le développement du projet mais n'avons pas utilisé de package.

Toujours dans la lignée de Sébastien, nous avons utilisé le même canevas de classes. De cette manière, chaque outil est composé d'au moins une méthode « setup(…) » qui permet de paramétrer l'outil et une méthode « run() » qui permet d'exécuter l'outil.

5.2 Types Principaux

Cette section vise à expliquer les principaux types d'objets créés tout au long du projet. Cela permettra au futur développeur de comprendre plus facilement l'action d'une méthode en regardant les types d'objets utilisés.

- Arguments : Structure contenant tous les paramètres pour l'algorithme de déformation
- Seed2D : coordonnées d'un point dans un plan 2D, utilisé pour poser une graine sur une coupe
- Seed3D : coordonnées d'un point dans un plan 2D, utilisé pour poser une graine sur un volume
- SeedSet2D : Ensemble de Seed2D
- SeedSet3D : Ensemble de Seed3D
- InputImageType : Image 2D en niveaux de gris fournie en entrée d'une méthode
- InputImageType3D : Volume en niveaux de gris fourni en entrée d'une méthode
- FilterImageType : Image 2D utilisée dans la plupart des filtres fournis par ITK
- FilterImageType3D : Volume utilisé dans la plupart des filtres fournis par ITK
- ImageIOType : Objet de liaison entre ITK et GDCM, pour la lecture et l'écriture d'image au format DICOM
- NamesGeneratorType : Générateur d'identifiant pour la création d'une nouvelle série DICOM

D'autres objets ont été créés mais ces derniers ne passent pas comme arguments. Ils représentent les implémentations d'algorithmes fournis par ITK. Ils sont donc utilisés au sein des méthodes afin d'effectuer un traitement sur les données. Nous ne les énoncerons pas ni ne les expliquerons.

5.3 Spécifications

Dans cette partie, nous donnerons une description des classes principales du projet ainsi que les spécifications de certaines de leurs méthodes.

5.3.1 ConnectedThreshold3DSegmenter

Cette classe représente l'outil de segmentation en 3D utilisant l'algorithme de croissance de région. La version de l'algorithme utilisé est appelée « ConnectedThreshold ».

`ConnectedThreshold3DSegmenter``()`

Crée un nouveau ConnectedThreshold3DSegmenter.

```
int setup(InputImageType3D::Pointer img, InputImageType3D::PixelType
lower, InputImageType3D::PixelType upper, SeedSet3D seeds)
```

Retourne 0 si l'image « img » à traiter, la borne inférieure « lower », la borne supérieure « upper » définissant l'intervalle d'intensité lumineuse caractérisant la région à segmenter et l'ensemble des marqueurs « seed » identifiant la région à segmenter ont été correctement spécifiés, 1 sinon.

```
int run()
```

Retourne 0 si le processus de segmentation s'est terminé correctement, 1 sinon.

`SeedSet3D` **`getSeed`**`()`

Retourne l'ensemble de graines utilisé pour la segmentation.

`InputImageType3D` **`getResult`**`()`

Retourne l'image binaire résultant de la segmentation telle que les pixels appartenant à la région d'intérêt ont la valeur 255 et ceux n'appartenant pas à la région d'intérêt la valeur 0.

5.3.2 ConnectedThresholdSegmenter

Cette classe représente l'outil de segmentation 2D par l'algorithme de croissance de région. Nous utilisons la même version de l'algorithme que pour la segmentation en 3D.

`ConnectedThresholdSegmenter``()`

Crée un nouveau ConnectedThresholdSegmenter.

```
int setup(InputImageType::Pointer img, InputImageType::PixelType lower,
InputImageType::PixelType upper, SeedSet3D seeds)
```

Retourne 0 si l'image « img » à traiter, la borne inférieure « lower », la borne supérieure « upper » définissant l'intervalle d'intensité lumineuse caractérisant la région à segmenter et l'ensemble des marqueurs « seed » identifiant la région à segmenter ont été correctement spécifiés, 1 sinon.

`int run``()`

Retourne 0 si le processus de segmentation s'est terminé correctement, 1 sinon.

`InputImageType3D` **`getResult`**`()`

Retourne l'image binaire résultant de la segmentation telle que les pixels appartenant à la région d'intérêt ont la valeur 255 et ceux n'appartenant pas à la région d'intérêt la valeur 0.

5.3.3 Controller

Cette classe est le noyau de l'application, c'est le contrôleur. C'est par cette classe que toutes les informations transitent. C'est aussi par cette classe que nous pouvons exécuter un outil.

`Controller``()`

Crée un nouveau Controller.

```
int setup(char* inputDirectory, char* outputDirectory, char*
atlasrepertory)
```

Configure le Controller, paramétrage des dossiers d'entrée et de sortie. Renvoie 0 si l'initialisation s'est bien passée, 1 sinon.

`int run``(`**`void`**`)`

Lance le viewer permettant la visualisation de la série, retourne 0 si le contrôle de la vue a bien été lancé, 1 sinon.

Int displayOverlay()

Retourne 0 si l'activation de l'affichage du résultat de la segmentation en overlay des coupes originales s'est terminée correctement, 1 sinon.

Int displayAtlas()

Retourne 0 si l'activation de l'affichage de l'atlas déformé sur les coupes originales s'est terminée correctement, 1 sinon.

Int displaySeed()

Retourne 0 si l'activation de l'affichage de l'ensemble des graines sélectionné pour la méthode 3D s'est terminée correctement, 1 sinon.

void initParameters()

Démarre la paramétrisation de la segmentation en y ajoutant une semence et en choisissant l'intensité de gris.

int launchSegmentation3D()

Lance la segmentation 3D, retourne 0 si la segmentation a bien été lancée, 1 sinon.

int launchSegmentation()

Lance la segmentation, retourne 0 si la segmentation a bien été lancée, 1 sinon.

int launchErasingTool()

Retourne 0 si l'outil de suppression de région a été lancé correctement, 1 sinon.

int launchSeedingTool()

Retourne 0 si l'outil de pose de graine a été lancé correctement, 1 sinon.

int displayMesh()

Retourne 0 si la tâche de rendu 3D simple a été lancée correctement, 1 sinon.

int computeVolume()

Retourne 0 et affiche le résultat dans la console si le calcul du volume de cartilage segmenté s'est terminé correctement, 1 sinon.

int computeStats()

Retourne 0 et affiche les résultats si le calcul des statistiques relatives au résultat de la segmentation s'est terminé correctement, 1 sinon.

int cancelSeed()

Permet d'annuler la pose de la dernière graine. Ne fonctionne que si la segmentation n'a pas encore été lancée.

void setParam(**int** low ,**int** up ,**float** fact ,arguments a ,**float** lev ,**float** lowerthresh ,**float** sig ,**int** min ,**int** max)

Initialise les paramètres de l'ensemble des algorithmes. Low et up sont pour l'algorithme de croissance de région ; lev, lowerthresh et sig sont pour l'algorithme de ligne de partage des eaux. « Arguments a » représente tous les paramètres de l'algorithme de déformation. Fact est utilisé comme coefficient de croissance. Min et max sont les tailles minimale et maximale d'une zone de la segmentation.

5.3.4 Convert3DMHDToDCM

Cet outil permet de convertir un volume au format MetaImage en une série d'images au format DICOM.

`Convert3DMHDToDCM``()`

Crée un nouveau convertisseur.

```
int run(std::string dicomOriginal, std::string filemhd, std::string
outputPath, std::string suffix)
```

Retourne 0 si la transformation s'est bien effectuée, 1 sinon.

5.3.5 DcmToMhdConverter

Cet outil fait l'inverse du précédent, c'est-à-dire qu'il convertit une série d'images DICOM en un volume au format MétaImage

`DcmToMhdConverter``()`

Crée une instance de la classe.

```
int setup(FileNamesContainer filenames,std::string outputFilename)
```

Retourne 0 si l'objet "filenames" contenant l'ensemble ordonné des chemins d'accès absolu aux fichiers DICOM composant la série ainsi que le chemin d'accès absolu du fichier MetaImage à écrire ont été corectement spécifiés, 1 sinon.

```
int setup(std::string directory, std::string outputFilename)
```

Retourne 0 si le répertoire "directory" contenant la série ainsi que le chemin d'accès absolu "outputfilename" du fichier MetaImage à écrire ont été correctement spécifiés, 1 sinon.

```
int run()
```

Retourne 0 si le fichier au format MetaImage a été correctement écrit, 1 sinon.

5.3.6 FenPrinc

Cette classe représente le GUI de l'application. Elle permet l'affichage de l'interface et transmet tout vers la classe « controller ». Ainsi l'affichage est séparé de « l'intelligence » du programme. Nous ne donnerons pas les spécifications des méthodes de cette classe car elles ne font que la liaison vers le Controller.

5.3.7 RegistrationSC

Cet outil permet de faire la déformation de l'atlas vers le volume représentant le genou à segmenter. Il utilise l'algorithme « Diffeomorphic Demons » introduit au chapitre 3.

`RegistrationSC``()`

Crée une nouvelle instance du déformateur.

`void DemonsRegistrationFunction``(arguments args)`

Lance la déformation de l'atlas et écrit le résultat sur le disque dur.

`int setup``(arguments args)`

Configure le déformateur.

5.3.8 SeedPicker

Cet outil permet soit de poser une graine sur une coupe, soit d'effectuer un zoom sur l'image. Cet outil est lié à la visualisation d'une série d'images.

`void Execute``(vtkObject *, `**`unsigned long`**` event, `**`void`**` *)`

Exécute l'observeur d'objet ITK permettant d'interagir à l'aide de la souris avec la fenêtre de visualisation des coupes.

5.3.9 SeedPropagator

Cet outil permet de propager les graines d'une coupe sur la suivante. Il est li à l'algorithme de segmentation en 2D.

`SeedPropagator``()`

Crée un nouveau SeedPropagator.

`int setup``(InputImageType::Pointer img)`

Retourne 0 si l'image « img » à partir de laquelle les graines doivent être calculées a été correctement spécifié, 1 sinon.

`int run``()`

Retourne 0 si le calcul des graines a été correctement réalisé, 1 sinon.

`SeedSet2D` **`getBestSeeds`**`()`

Retourne l'ensemble des graines calculées si le calcul de celles-ci a été correctement réalisé.

`SeedSet3D` **`GetSeed3DFromAtlas`**`(InputImageType3D::Pointer atlas)`

Renvoie la liste des graines 3D sélectionnées sur l'atlas.

5.3.10 Segmenter

Cette classe est le manager de la segmentation en 2D. Elle permet de combiner les deux outils de segmentation 2D et de corriger les résultats. C'est elle aussi qui rajoute l'information sur l'intensité de gris.

`Segmenter::Segmenter()`

Crée une nouvelle instance de la classe.

```
int setup(std::string dir1, std::string dir2, std::string dir3,FenPrinc*
window)
```

Retourne 0 si le répertoire source contenant la série d'images au format DICOM à traiter et le répertoire de destination des fichiers résultant de la segmentation ont été correctement spécifiés, 1 sinon.

```
void initResultFiles(FileNamesContainer filenames)
```

Crée les fichiers qui contiendront les résultats.

```
int addRegion(std::string file, InputImageType::PixelType lower,
InputImageType::PixelType upper, SeedSet2D set, std::vector<Segment*>
boundaries,float level, float lowerThreshold , float sigma)
```

Retourne 0 si la région présente sur l'image DICOM spécifiée par « file » et caractérisée par l'intervalle d'intensité [lower,upper] et les graines « seeds » a été correctement segmentée et ajoutée à la région segmentée préalablement sur cette même coupe si elle existe, 1 sinon.

```
int cancelLastSeed()
```

Permet d'annuler la pose de la dernière graine. Ne fonctionne que si la segmentation n'a pas encore été lancée.

```
int eraseRegionInside(std::string file, IrregularPolygon* polygon)
```

Retourne 0 si la région segmentée sur l'image DICOM spécifiée par « file » et contenue à l'intérieur d'un polygone quelconque spécifié par « polygon » a été effacée correctement, 1 sinon.

```
int run(FileNamesContainer fileNames, int ind, InputImageType::Pointer
prev, InputImageType::PixelType lower, InputImageType::PixelType upper,
SeedSet2D seeds, float factor,float level, float lowerthreshold, float
sigma)
```

Retourne 0 si le processus de segmentation s'est terminé correctement, 1 sinon. L'objet « filenames » spécifie la séquence ordonnée d'images au format DICOM à traiter. Au sein de cette séquence, seules les coupes d'indices supérieurs ou égaux à « ind » doivent être traitées. L'image « previous » spécifie l'image binaire à utiliser pour détecter et corriger une fuite sur la première coupe de la série à traiter. Le paramètre « factor » spécifie le rapport maximum autorisé entre les surfaces segmentées sur deux coupes consécutives.

```
void resultCooperation(InputImageType::Pointer&
img,InputImageType::Pointer threshold, InputImageType::Pointer water,
InputImageType::Pointer imDep, int upper){
```

Utilise le résultat des deux algorithmes de segmentation afin de donner une meilleure segmentation.

`InputImageType::Pointer` **`Segmenter::correctImage`**`(InputImageType::Pointer img)`

Corrige le résultat de la segmentation en enlevant les zones trop petites et les zones trop grandes.

`void numberOfPoint``(SeedSet2D& set,InputImageType::Pointer& img,` **`unsigned int`**`& i,` **`unsigned int`**`& j,` **`int`**`& cpt)`

Renvoie le nombre de pixels contenus dans la zone (appel récursif).

5.3.11 SerieViewer

Cette classe permet de visualiser une série d'images ainsi que l'atlas déformé et la zone segmentée. Elle est incorporée dans la fenêtre principale et c'est par son intermédiaire et l'utilisation de la classe « SeedPicker » que nous pouvons poser des graines ou effectuer des zooms.

`SerieViewer``()`

Crée une instance de la classe.

`int setup``(`**`char`**`* inputDirectory,` **`char`**`* outputDirectory)`

Retourne 0 si l'initialisation de la vue s'est correctement terminée, 1 sinon.

`void setSize``()`

Spécifie la taille de la fenêtre en fonction de la taille de l'image.

`Int run``()`

Retourne 0 si la vue a été correctement lancée, 1 sinon.

`void displayInfos``()`

Affiche à l'écran les informations du fichier DICOM.

`int clearOverlay``()`

Désactive l'affichage du résultat de la segmentation en overlay des coupes originales.

`int displayOverlay``()`

Active l'affichage du résultat de la segmentation en overlay des coupes originales.

`void NextSlice``()`

Affiche la coupe suivante dans l'ordre anatomique des coupes.

`void PreviousSlice``()`

Affiche la coupe précédente dans l'ordre anatomique des coupes.

`int reverseSerie``()`

Inverse l'ordre d'affichage des coupes de la série.

`InputImageType3D* getImage()`

Retourne l'image tridimensionnelle formée par l'empilement des coupes successives de la série.

`int getCurrentSlice()`

Retourne le numéro (l'indice) de la coupe courante.

`void Zoom(double amount)`

Une valeur de « amount » > 1 produit un agrandissement. Une valeur de « amount » < 1 produit un rétrécissement.

`int clearAtlas()`

Désactive l'affichage de l'atlas.

`int displayAtlas()`

Active l'affichage du résultat de la déformation de l'atlas.

`int clearSeed()`

Désactive l'affichage des graines.

`int displaySeed()`

Active l'affichage des graines sélectionnées pour la méthode 3D.

5.3.12 StatsComputer

Cet outil permet de calculer diverses statistiques sur la région d'intérêt. Parmi ces statistiques, nous retrouvons le minimum d'intensité, le maximum d'intensité, la moyenne et l'écart type.

`StatsComputer(void)`

Crée une nouvelle instance de la classe.

`int setup(InputImageType::Pointer img, InputImageType::Pointer msk)`

Retourne 0 si l'image binaire « msk » décrivant la région segmentée (telle que les pixels appartenant à la région d'intérêt ont la valeur 255 et ceux n'appartenant pas à la région d'intérêt la valeur 0) sur l'image « img » a été correctement spécifiée, 1 sinon.

`int run(void)`

Retourne 0 si le calcul des statistiques (valeurs d'intensité maximum, minimum, moyenne, écart-type et variance) a été correctement réalisé, 1 sinon.

`InputImageType::PixelType getMin()`

Retourne la valeur d'intensité minimale des pixels segmentés sur l'image.

`InputImageType::PixelType` **`getMax`**`()`

Retourne la valeur d'intensité maximale des pixels segmentés sur l'image.

`int getCount``()`

Retourne le nombre de pixels segmentés sur l'image.

`float getAverage``()`

Retourne la moyenne des valeurs d'intensité des pixels segmentés sur l'image.

`float getStdDeviation``()`

Retourne l'écart-type des valeurs d'intensité des pixels segmentés sur l'image.

`float getVariance``()`

Retourne la variance des valeurs d'intensité des pixels segmentés sur l'image.

5.3.13 VolumeComputer

Cet outil permet de calculer la taille en mm³ d'un volume. L'image doit être fournie sous le format MetaImage.

`VolumeComputer``(`**`const char`** `* rawFile,` **`const char`** `* mhdFile)`

Crée une nouvelle instance de la classe.

`int computeSize``()`

Retourne la taille du volume.

`float computeVoxelVolume``()`

Retourne le volume d'un voxel[16].

`int countVoxelNumber``()`

Retourne le nombre de voxels que contient le volume.

`int getResult``()`

Retourne le volume de l'objet 3D présent dans l'image de sorte que les voxels appartenant à celui-ci ont la valeur 255 et ceux ne lui appartenant pas la valeur 0.

5.3.14 VolumeCorrecter

Cet outil permet de corriger un volume en utilisant l'atlas déformé.

`VolumeCorrecter``()`

Crée une nouvelle instance de la classe.

[16] Un voxel est un pixel en trois dimensions.

```
int setup(float fact, InputImageType3D::Pointer ref,
InputImageType3D::Pointer& volume, InputImageType3D::Pointer water,
InputImageType3D::Pointer depart , int up, int min, int max)
```

Retourne 0 si l'image binaire « curr » à corriger, l'image binaire « prev » à partir de laquelle la correction doit être effectuée et le rapport « fact » maximal autorisé entre les surfaces segmentées sur les deux images binaires ont été correctement spécifiés, 1 sinon.

```
int run()
```

Retourne 0 si la correction a été correctement réalisée, 1 sinon.

```
Bool detectOverSegmentation()
```

Retourne « true » si l'image binaire « curr » présente un défaut de segmentation par rapport à l'image binaire « prev », « false » sinon.

```
InputImageType3D::Pointer correctImage(InputImageType3D::Pointer img)
```

Corrige le résultat de la segmentation en enlevant les zones trop petites et les zones trop grandes

```
void numberOfPoint(SeedSet3D& set,InputImageType3D::Pointer& img, unsigned
int& i, unsigned int& j,unsigned int k, int& cpt)
```

Renvoie le nombre de pixels contenus dans la zone (appel récursif)

5.3.15 VolumeResample

Cet outil permet d'adapter la taille des pixels de l'atlas à la taille des pixels de la série d'images représentant le genou à segmenter. Sans cet outil, il serait impossible de faire une correspondance coupe par coupe.

```
int VolumeResample::setup(InputImageType3D::Pointer
current,InputImageType3D::Pointer atlas, std::string path)
```

Retourne 0 si il a bien été initialisé, 1 sinon.

```
int VolumeResample::run ()
```

Retourne 0 si la mise à l'échelle des pixels a été correctement réalisée, 1 sinon.

5.3.16 VolumeViewer

Cette classe représente l'outil de visualisation du volume en 3D. Sa fenêtre est incorporée dans la fenêtre principale.

```
VolumeViewer()
```

Crée une nouvelle instance de la classe.

```
int setup(InputImageType3D* img)
```

Retourne 0 si l'image 3D « img » à visualiser a été correctement spécifiée, 1 sinon.

void Display(vtkPolyData* polyData)

Lance l'affichage du rendu 3D du volume de cartilage segmenté. « polydata » représente le volume de cartilage

int run(**void**)

Retourne 0 si le rendu de l'image 3D a été correctement réalisé, 1 sinon. Remarque : rendu surfacique par l'algorithme des Marching Cubes.

vtkRenderWindowInteractor* **getInteractor**()

Retourne l'interactor du rendu 3D.

5.3.17 WatershedSegmenter

Cet outil représente l'outil de segmentation 2D utilisant l'algorithme de la ligne de partage des eaux.

WatershedSegmenter()

Crée un nouveau WatershedSegmenter.

LabeledImageType* **getResult**()

Renvoie le résultat de la segmentation avec l'algorithme de la ligne de partage des eaux.

int setup(std::string img, **float** level, **float** lowerThreshold , **float** sigma)

Retourne 0 si les paramètres ont été correctement spécifiés, 1 sinon.

int run()

Retourne 0 si le processus de segmentation s'est terminé correctement, 1 sinon.

5.4 Diagrammes de classes

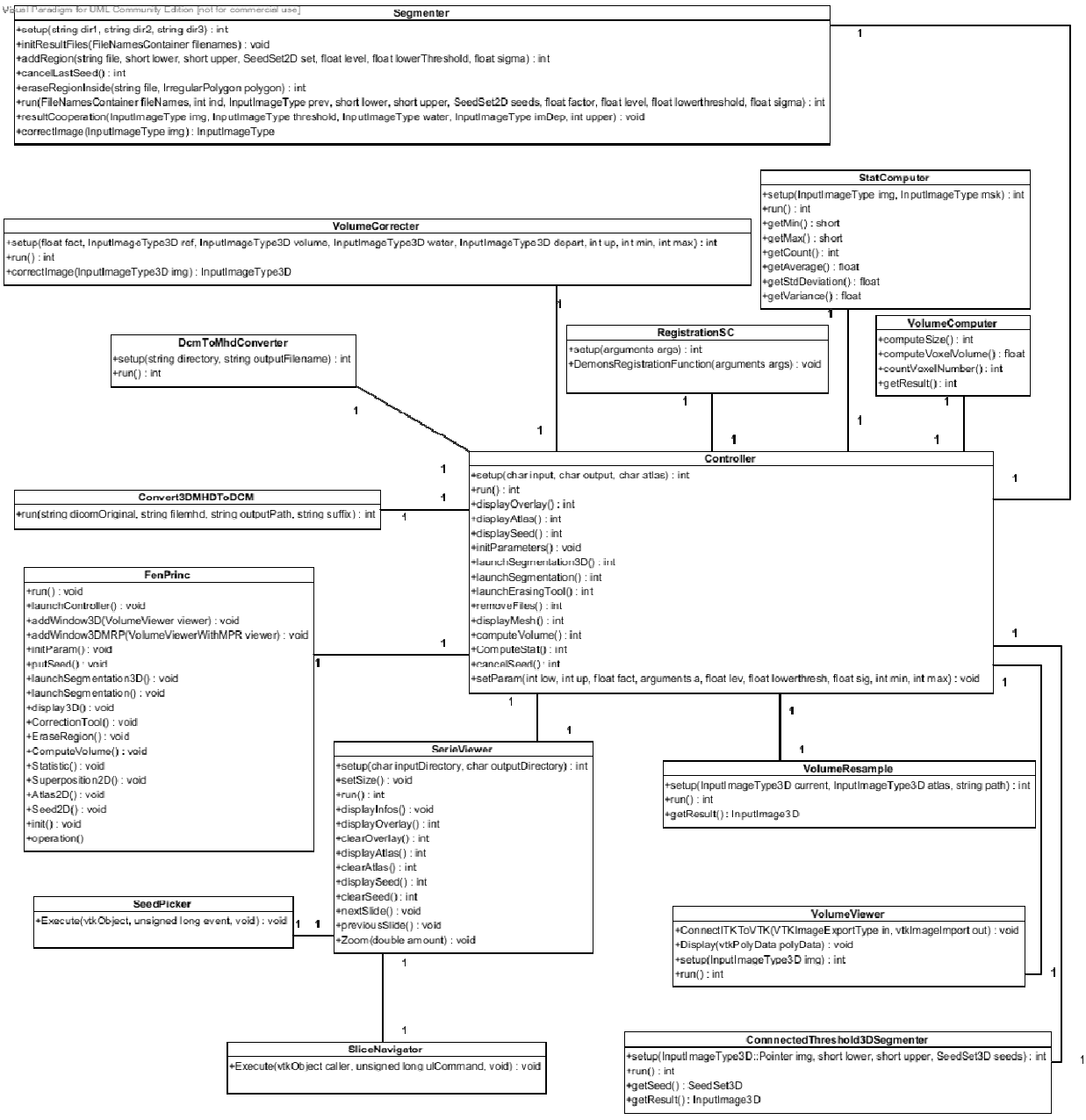


Figure 49 - Diagramme de classes principal

Le Controller est le noyau de l'application, c'est autour de lui que viennent se greffer les autres outils. Toutefois, l'utilisateur interagi avec le controller de façon indirecte par l'intermédiaire de l'interface graphique représentée par la classe « FenPrinc ».

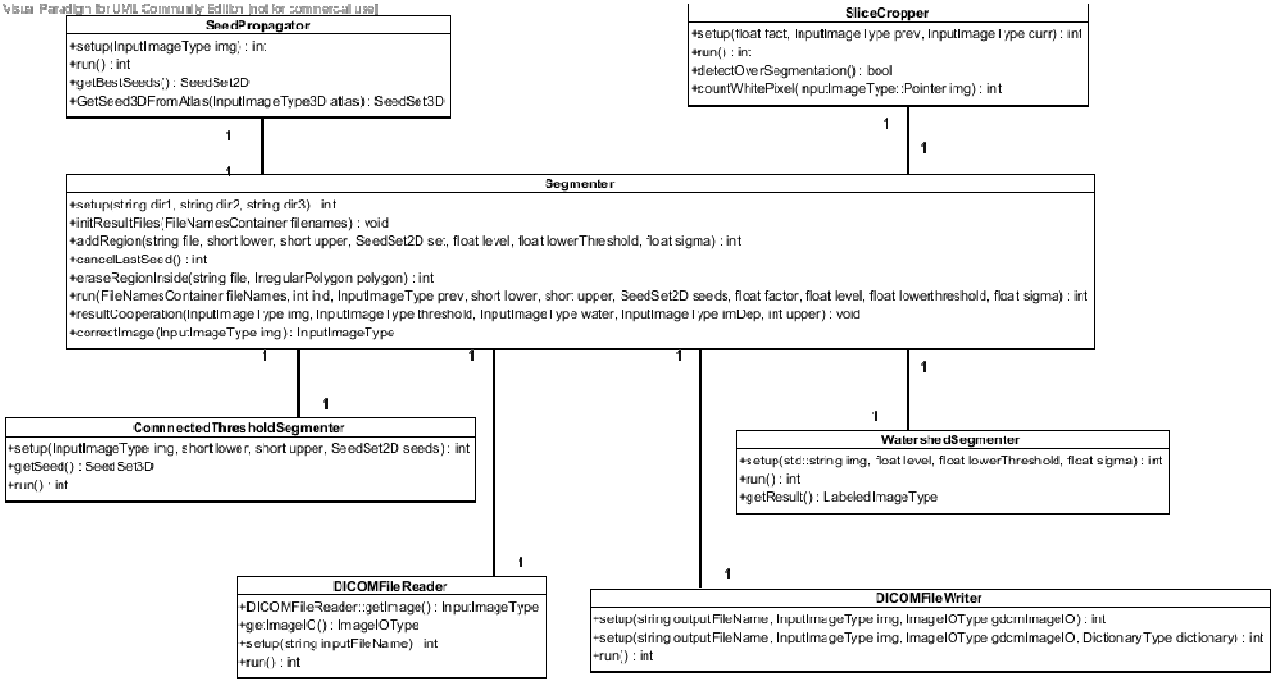


Figure 50 - Diagramme de classes de l'outil de segmentation 2D

Le Segmenter est l'administrateur de la segmentation en 2D. On ne retrouve pas ce genre d'administrateur pour la segmentation 3D car au terme du stage cette technique était toujours en cours de développement.

5.5 Diagrammes de séquence

Les diagrammes de séquences suivants permettent de connaître le fonctionnement de l'application. Bien que les passages de données ne soient pas mentionnés, on en a une idée par l'ordre des traitements effectués. Les classes destinées à la visualisation ne sont pas indiquées dans les diagrammes mais leurs traitements de données sont greffés au controller et leurs fonctions de visualisation incorporées dans la classe FenPrinc.

Le premier diagramme concerne le lancement de la segmentation 2D. Il reprend les étapes importantes de lancement d'objet, de paramétrisation et de traitements. Dans ce diagramme, l'utilisateur n'effectue pas de corrections manuelles. Il s'agit du cas « best scenario », autrement dit le cas idéal.

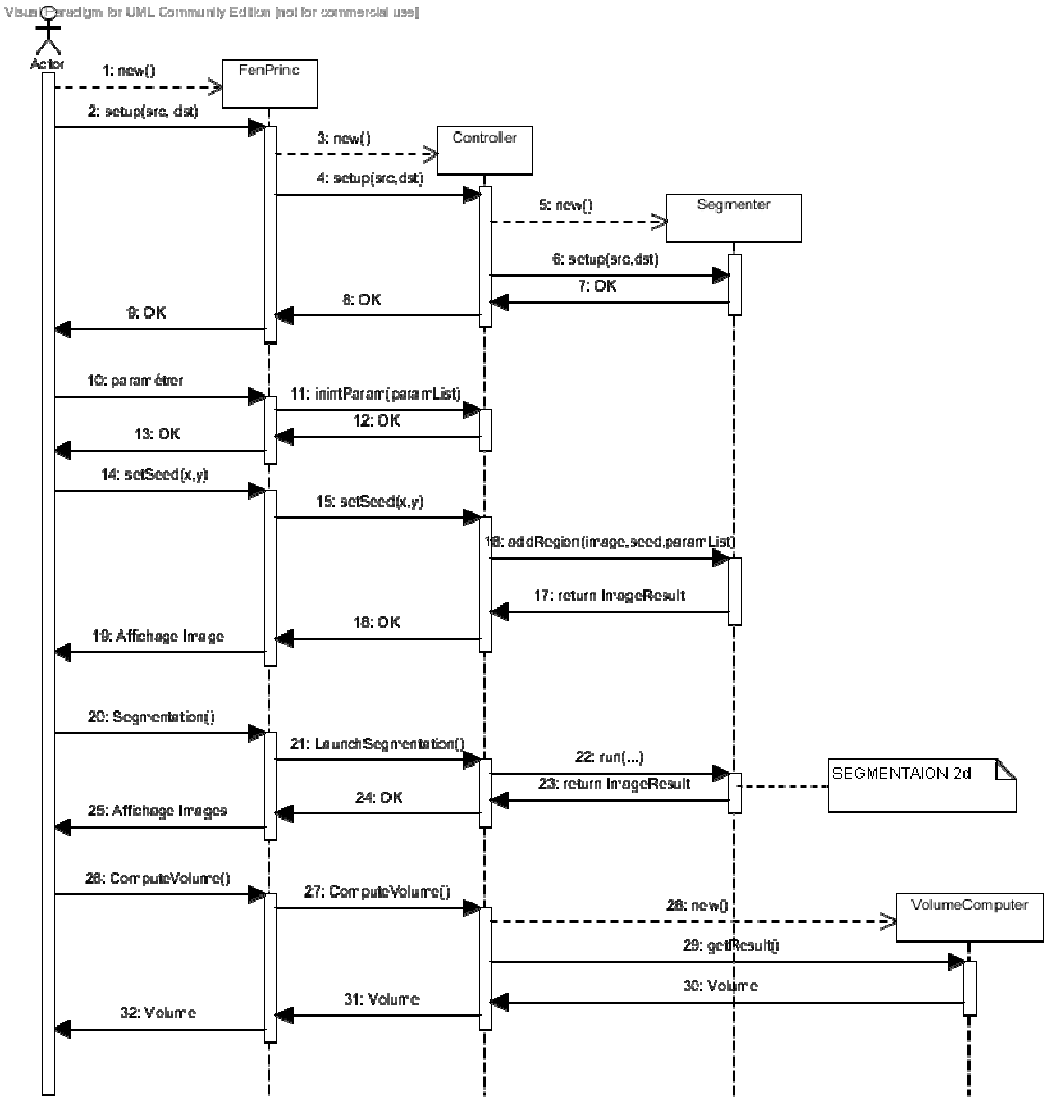
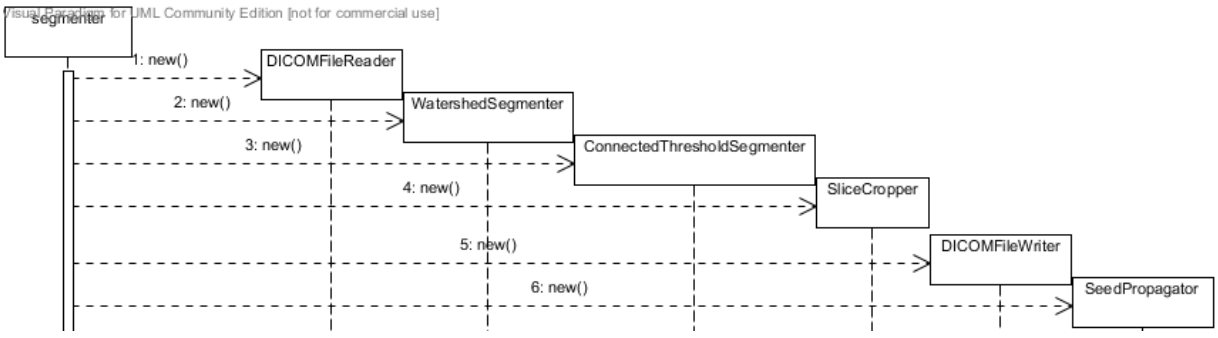


Figure 51 – Diagramme de séquence du lancement de la segmentation 2D

Le deuxième diagramme détaille le processus de segmentation 2D. Il est le détail de diagramme précédent au niveau de la note « Segmentation 2d ».



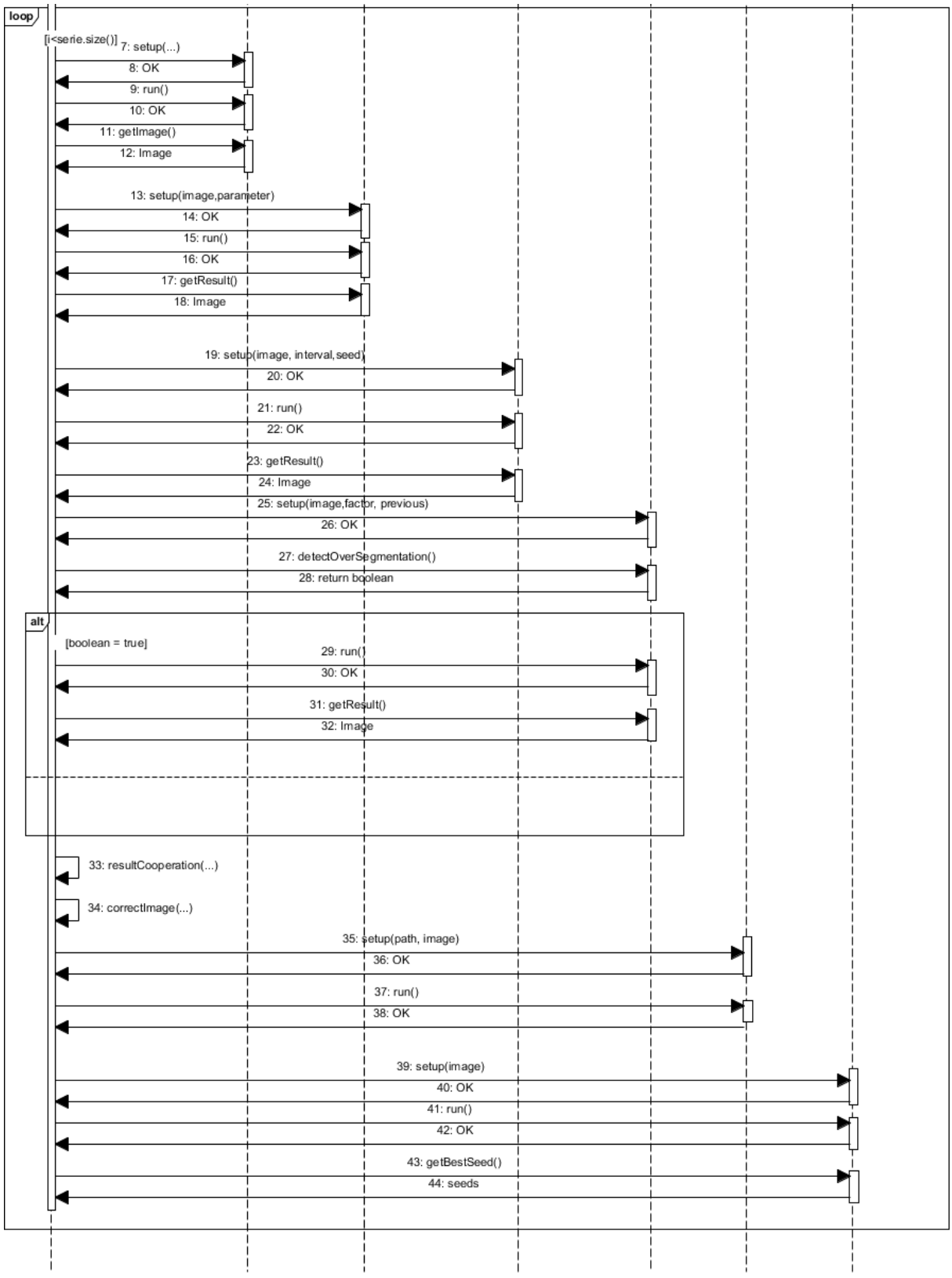
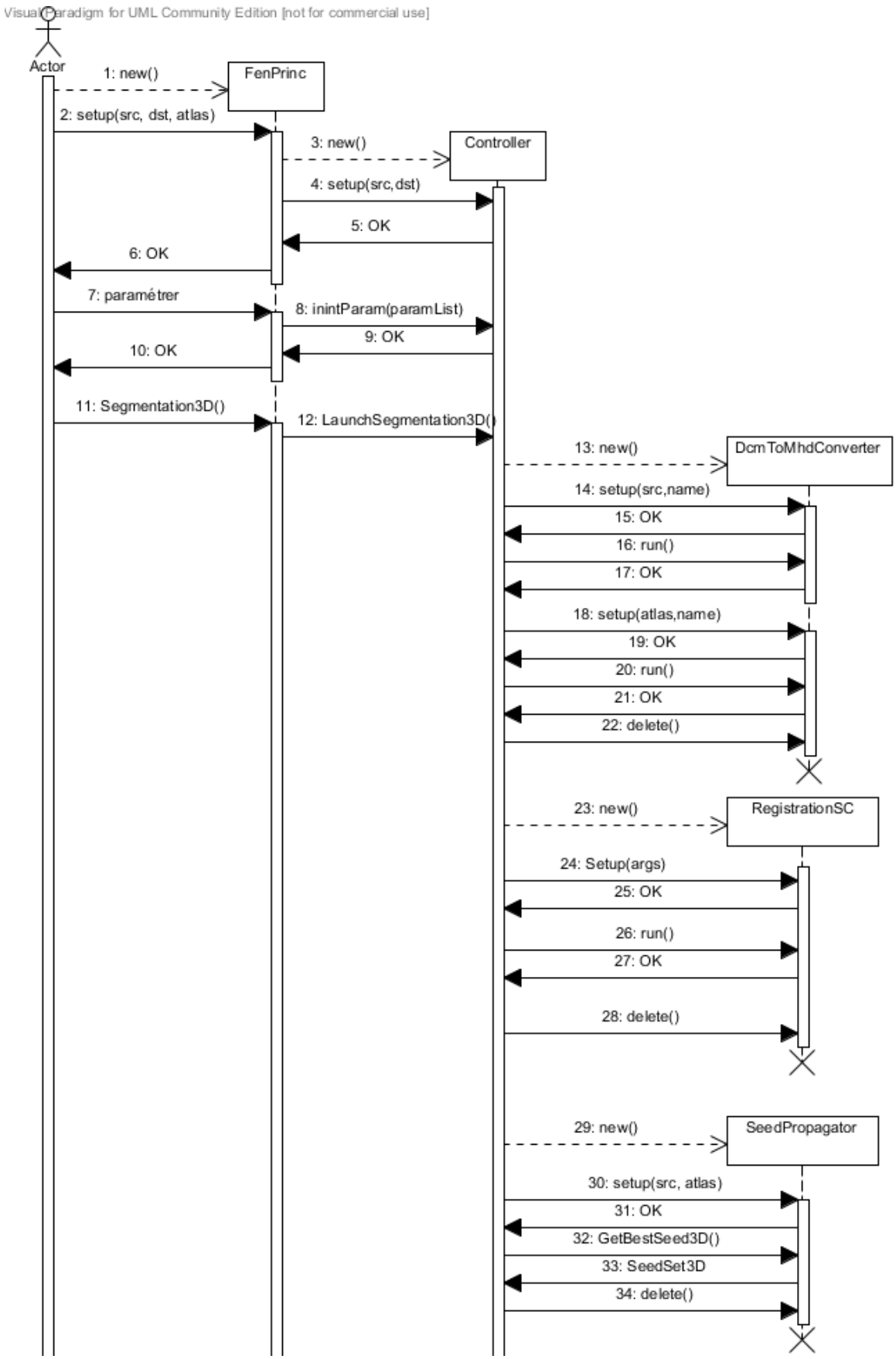


Figure 52 – Diagramme de séquence du processus de segmentation 2D

Le troisième diagramme illustre le fonctionnement de processus de segmentation par la méthode 3D. Encore une fois, nous ne donnons pas le cas où l'utilisateur effectue des corrections manuelles.


Visual Paradigm for UML Community Edition [not for commercial use]
Actor
FenPrinc
Controller
DcmToMhdConverter
RegistrationSC
SeedPropagator
1: new()
2: setup(src, dst, atlas)
3: new()
4: setup(src,dst)
5: OK
6: OK
7: paramétrer
8: inintParam(paramList)
9: OK
10: OK
11: Segmentation3D()
12: LaunchSegmentation3D()
13: new()
14: setup(src,name)
15: OK
16: run()
17: OK
18: setup(atlas,name)
19: OK
20: run()
21: OK
22: delete()
23: new()
24: Setup(args)
25: OK
26: run()
27: OK
28: delete()
29: new()
30: setup(src, atlas)
31: OK
32: GetBestSeed3D()
33: SeedSet3D
34: delete()

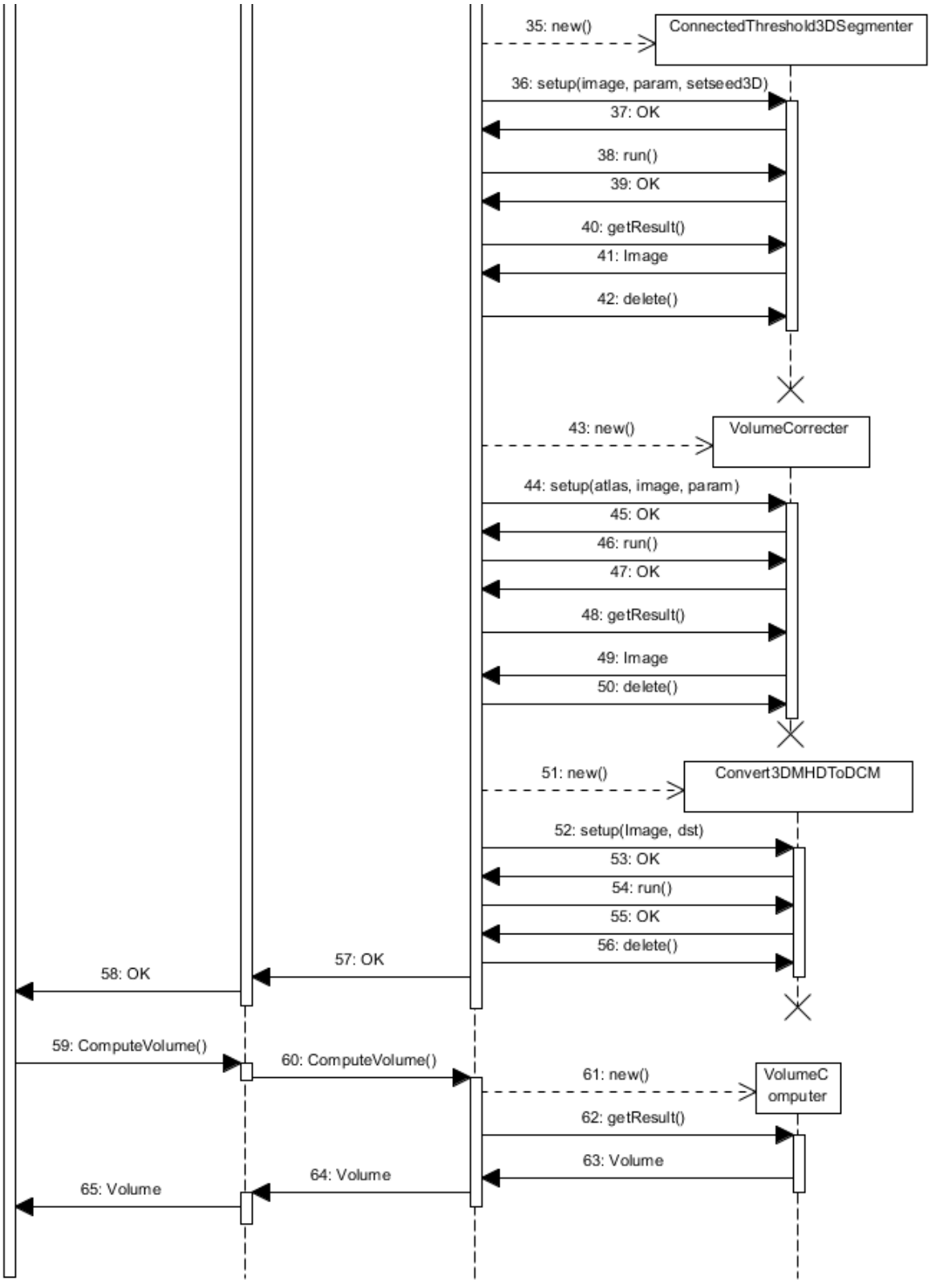


Figure 53 - Diagramme de séquence du processus de segmentation 3D

5.6 Intégration dans Telemis

L'intégration de l'application dans Telemis se fait par l'introduction d'un plugin dans Telemis. Nous allons maintenant décrire la méthode de création d'un plugin pour Telemis.

1- Créer un fichier d'extension CFG : mon_plugin.cfg
2- Déplacer le fichier dans le répertoire suivant :
c:\Program Files\telemis\telemis\entity\tmrhe\cfg\imageProcessingPlugins
3- Éditer le fichier comme suit :

```
outputPath = C:/fromTelemis
outputFormat = DICOM
thirdPartySoftware = C:/../launcher.bat
thirdPartySoftwareName = Nom_Plugin
inputPath = C:/toTelemis
inputFormat = DICOM
inputExtension = DCM
okExitCode = 0
```

Le champ outputPath de Telemis correspond à l'input de notre application et inversement avec le champ inputPath.

4- Créer un fichier d'extension .bat : lancher.bat
5- Éditer le fichier comme suit :

Insérer la ligne suivante :

C:\CHEMIN_COMPLET\plugin.exe arg1 … argN

Remarque : pour que l'invite de commande s'affiche il faut ajouter, au début de la ligne, le mot « start ».

Chapitre 6 – Discussion

Ce chapitre a pour but de présenter des pistes d'amélioration pour l'application. Les améliorations proposées porteront principalement sur le temps d'exécution et la qualité des résultats. Le chapitre sera divisé en trois grandes parties, la première axée sur le processus de segmentation et la deuxième axée sur les données. À cela se rajoute une partie sur l'interface.

6.1 Amélioration du processus

6.1.1 Application multithreads

Les processus de segmentation sont actuellement exécutés de manière séquentielle. Cela s'explique par le fait que l'application n'utilise qu'un seul thread. Le développement de l'application vers l'utilisation de plusieurs threads permettrait d'utiliser les processeurs multi-core de manière optimale. En effet, pour le moment l'application n'utilise qu'un seul cœur dans sa totalité.

Cette restructuration de l'application permettrait pour le processus de segmentation 2D de s'exécuté de manière plus rapide. La segmentation par croissance de région se ferait alors en même temps que la segmentation par ligne de partage des eaux.

6.1.2 Coopération de méthode

Nous avons énoncé les principes de la coopération de méthodes au chapitre trois. Nous avons montré que plusieurs coopérations étaient possibles. L'application utilise pour le moment L'application utilise la coopération de résultat. Bien que la méthode utilisé ait permis d'amélioré les résultats finaux, elle a aussi augmenté le temps de traitement de manière significative.

Comme nous l'avons expliqué dans le chapitre 4, ce sont les traitements appliqués aux résultats des algorithmes qui demandent le plus de temps. Bien que l'algorithme de la ligne de partage des eaux permette d'agrandir la zone segmentée par l'algorithme de croissance de région, il n'empêche pas les fuites qui demandent alors un grand travail de correction.

Il a été montré que les méthodes de segmentation par croissance de région et par contour fonctionnaient très bien en coopération. La méthode de contour permettant de donner des informations concernant la zone d'intérêt à l'algorithme de croissance de région [15].L'ajout de cette coopération à l'application permettrait sans aucun doute d'empêcher certaines fuites, ce qui réduira le temps de correction.

6.1.3 L'atlas

Nous avons vu au chapitre 3 les façons d'utiliser un atlas. La méthode choisie pendant le développement est la première méthode (un atlas mono-résultat). L'utilisation de la méthode la plus simple s'explique par le fait que cette méthode à commencé à être développée en fin de période.

Afin d'améliorer les résultats de la méthode de segmentation 3D, l'utilisation de l'atlas doit être améliorée. Une façon simple de le faire serait de prendre l'atlas le plus ressemblant dans l'ensemble possible. ITK fournit un algorithme de calcul de différence entre image, il est d'ailleurs

utiliser dans la méthode de déformation afin de constater un rapprochement des deux images. En plus d'améliorer les résultats, cela diminuerait le temps de déformation.

Il est bien sur possible d'utiliser une des deux autres méthodes (moyenne des atlas et multi-atlas). Dans tous les cas, l'ensemble des atlas possible devra être étoffé afin de garantir un meilleur résultat.

6.2 Amélioration des données

Pour l'amélioration des images à traiter, plusieurs procédures sont possibles, comme la combinaison avec des images d'un autre type d'acquisition ou l'augmentation de la résolution. Toutefois nous n'envisagerons pas ces possibilités. Dans le premier cas, il est envisageable de d'abord essayer l'application tel quelle sur le nouveau type d'acquisition avant de combiner plusieurs type d'acquisition comme l'a dit Sébastien. Pour l'autre possibilité, le problème se situe dans le bruit contenu dans le signal qui permet de créer l'image. En augmentant la taille de l'image on essaie de capter plus de détails qui sont difficilement repérable dans le signal.

6.2.1 Filtre

Une difficulté majeure de la segmentation du cartilage est la finesse de ce dernier. De plus, il n'y a qu'une petite différence de contraste avec les autres tissus dit mous. L'application d'un filtrage sur l'image pourrait rendre plus décelable la différence entre cartilage et muscle et ainsi diminuer le phénomène de fuite lors de la segmentation. Ces filtres vont retirer une partie du bruit présent dans l'image et homogénéiser les couleurs, il sera possible alors de mieux cibler l'intervalle de niveaux de gris pour ne prendre que le cartilage.

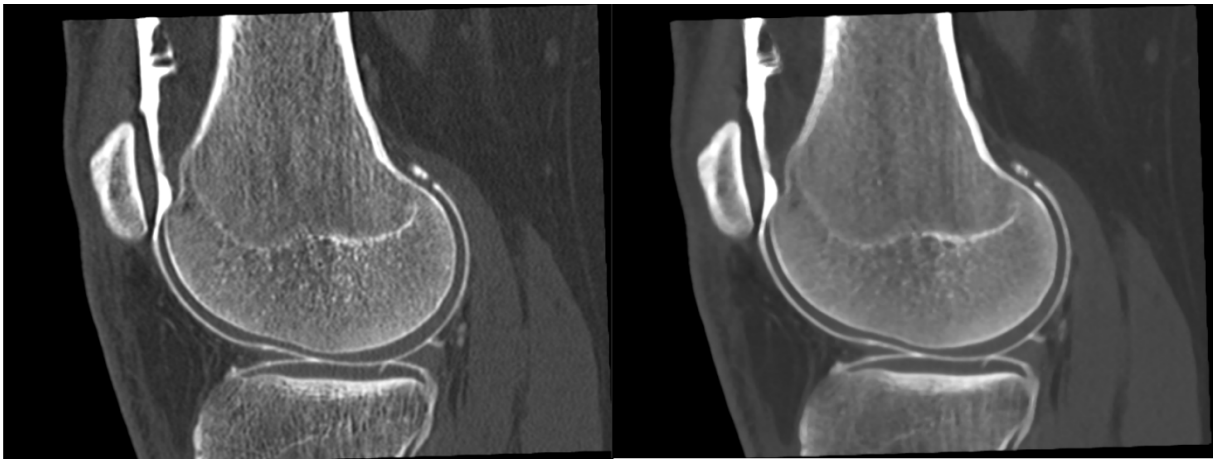


Figure 54 - À gauche l'image originale ; À droite l'image lissée par un filtrage médian

ITK fournit un ensemble de filtres permettant d'augmenter la qualité des images. Toutefois ces filtres demandent un changement d'encodage de l'image. Des étapes de transformations supplémentaires sont donc demandées afin de garder les images compatibles avec les algorithmes de segmentation.

6.3 Amélioration de l'interface

À ce point du développement, l'interface présente encore quelques lacunes comme l'absence de raccourcis clavier, l'interdiction d'utiliser les croix de fermeture des fenêtres et la perte d'affichage de l'interface pendant les traitements. Régler ces problèmes permettrait d'améliorer de façon conséquente l'utilisation de l'application.

En plus de ces lacunes, une amélioration de l'outil de zoom est nécessaire. En effet à ce stade, l'outil ne permet pas de dépasser la taille de la fenêtre. Si le zoom est trop important la partie supérieure et celle de droite ne sont plus visibles car hors du cadre de la fenêtre. L'ajout de barres de navigation à la fenêtre devrait régler le problème.

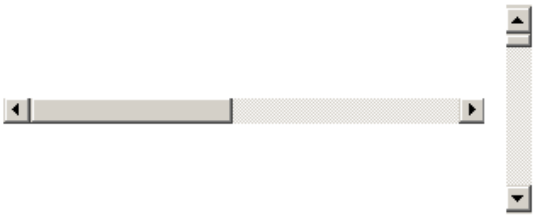


Figure 55 - Barres de navigation de fenêtre

Une dernière amélioration de l'interface est l'ajout d'une mesure de l'épaisseur du cartilage dans le rendu 3D. Pour le moment, ce rendu ne possède aucune information additionnelle et représente le volume en vert. L'information peut être ajoutée par un code couleur et une échelle.

Figure 56 - Échelle de couleur

Conclusion

Travailler dans le domaine de l'imagerie médicale est passionnant. Bien que difficile à prendre en main et nécessitant du temps pour en maitriser les principaux aspects, le travail d'informaticien dans ce domaine est très valorisant.

La recherche de solutions pour le problème de la segmentation du cartilage du genou est un sujet d'étude intéressant et stimulant. Les résultats obtenus avec l'application développée durant le stage, bien que nécessitant des améliorations, sont prometteurs. Les deux méthodes de segmentation proposées ont abouti à des stades de développement différents et la qualité de leurs résultats sont en correspondance avec leur développement.

En effet, la méthode de coopération d'algorithme a permis d'améliorer les résultats obtenus par la version précédente de l'application, en épousant un peu plus les bords du cartilage. La méthode 3D utilisant l'atlas a demandé plus de recherche pour sa mise en place, ce concept n'ayant pas été étudié par mon prédécesseur et étant plus avancé dans le domaine de la segmentation d'image. Bien que les résultats obtenus par cette méthode soient très différents en fonction du patient et de l'atlas choisi, c'est cette méthode qui devra à l'avenir être au centre de la recherche.

De manière générale, les objectifs fixés au début de ce travail ont été remplis. En effet, l'ajout de l'interface graphique et de divers outils rend l'utilisation de l'application plus aisée. De plus, étant donné que plusieurs problèmes ont été réglés, l'application souffre moins d'arrêts non prévus.

Même si l'objectif principal lié à la segmentation du cartilage a été rempli, le lecteur aura remarqué que le sujet est loin d'être clos. En effet, diverses perspectives de recherche ont été présentées dans ce mémoire.

Pour conclure, il faut insister sur le fait que le milieu médical est un secteur très valorisant pour l'informaticien. Apporter son concours pour améliorer le bien-être des gens est une cause qui ne laisse personne de marbre.

Par ailleurs, se retrouver dans un milieu totalement nouveau est une expérience très enrichissante. En effet cette nouveauté permet à l'informaticien d'améliorer une grande exigence qui leur est souvent requise dans le métier : Comprendre l'environnement et se faire comprendre de lui pour lui fournir des services de qualité.

Annexe

FUNDP
INSTITUT D'INFORMATIQUE
Rue Grandgagnage, 21
5000 NAMUR (BELGIUM)

CLINIQUES UNIVERSITAIRES
MONT-GODINNE
Avenue du Docteur Gaston Thérasse, 1
5530 YVOIR (BELGIUM)

MEDECINE NUCLEAIRE

Médecine Nucléaire : Amélioration d'un outil de traitement pour la segmentation d'images médicales 2D et 3D dans le domaine de l'ostéoarticulaire

Michiels Yvan

Promoteur : Jean-Paul LECLERCQ
Co-promoteur : Hubert MEURISSE

GUIDE D'UTILISATION

Année académique 2011-2012
Mémoire présenté en vue de l'obtention du grade de Master en Sciences Informatiques

Table des matières

1 Compilation

Cette partie aborde le sujet de la compilation du projet. Utilisant différentes librairies, le projet est assez difficile à compiler. En effet, l'utilisation de plusieurs outils est nécessaire afin de bien installer les librairies utilisées par le logiciel. Cette partie sera donc très intéressante pour toute personne désireuse de améliorer le programme.

La procédure de compilation se veut assez générique car seuls des logiciels open source seront utilisés. La procédure a été effectuée avec le logiciel de développement C++ Eclipse.

1.1 Préparation

Voici les logiciels et librairies nécessaires pour le projet ainsi que la configuration de l'environnement de développement :

- Télécharger et installer CMake, Msys, MinGW, Eclipse C++ (Wascana) et Qt (librairie d'interface graphique).

- Modifier la variable d'environnement « PATH » en y ajoutant le chemin d'accès au répertoire bin de CMake, de Msys, de MinGW, de Qt et d'Eclipse C++.

- Télécharger les librairies VTK et ITK.

Remarque : Msys n'est utilisé que pour son invite de commande. Il est possible de passer par l'invite de commande de Windows.

1.2 ITK

Voici la procédure d'installation de la librairie d'ITK :

- Extraire les sources d'ITK dans un répertoire, par exemple « c:\tmp\ITK ».

- Dans le répertoire des sources, créer un répertoire, par exemple « binary » (il contiendra les fichiers compilés).

- Lancer CMake, ajouter le chemin vers les sources (ici « c:\tmp\ITK ») et le chemin pour les binaires (ici « c:\tmp\ITK\binary »).

- Presser le bouton « CONFIGURE ».

- Choisir le générateur « MSYS Makefiles » cocher « Specify native compilers » et passer à l'écran suivant.

- Spécifier le compilateur C utilisé par Eclipse (ou votre logiciel de développement) ainsi que le compilateur C++ et presser « FINISH ».

- Spécifier, dans le champ « CMAKE_INSTALL_PREFIX », le chemin où sera installé ITK, par exemple « c\Program Files\ITK ».

- Cocher la case « ADVANCED » pour la recherche et décocher « PROJ_USE_PTHREADS ».

- Cocher « ITK_USE_REVIEW ».

- Appuyer sur « CONFIGURE ».

- Appuyer sur « GENERATE ».

- Lancer « msys.bat » (invite de commande simulant linux), se rendre dans le répertoire des binaires (ici « c:\tmp\ITK\binary »)[17].

- Toujours dans l'invite de commande de Msys, taper « make » afin de lancer l'installation d'ITK, l'opération peut durer plusieurs heures.

- ITK est maintenant installé, il faut ajouter à la variable d'environnement « PATH » le chemin d'accès au répertoire bin d'ITK (ici « c:\Program Files\ITK\bin »).

1.3 VTK

Voici la procédure d'installation de la librairie d'VTK :

- Extraire les sources d'VTK dans un répertoire, par exemple «c:\tmp\VTK».

- Dans le répertoire des sources, créer un répertoire, par exemple «binary» (il contiendra les fichiers compilés).

- Lancer CMake, ajouter le chemin vers les sources (ici «c:\tmp\VTK») et le chemin pour les binaires (ici «c:\tmp\VTK\binary»).

- Presser le bouton «CONFIGURE».

- Choisir le générateur «MSYS Makefiles», cocher «Specify native compilers» et passer à l'écran suivant.

- Spécifier le compilateur C utilisé par Eclipse (ou votre logiciel de développement) ainsi que le compilateur C++ et presser «FINISH».

- Spécifier, dans le champ «CMAKE_INSTALL_PREFIX», le chemin où sera installé VTK, par exemple «c:\Program Files\VTK».

- Cocher la case «ADVANCED» pour la recherche et décocher «PROJ_USE_PTHREADS».

[17] Dans ce cas il faut taper la ligne de commande « cd /c/tmp/ITK/binary ».

- Cocher «VTK_USE_QT».
- Appuyer sur «CONFIGURE».
- Appuyer sur «GENERATE».
- Lancer «msys.bat» (invite de commande simulant linux), se rendre dans le répertoire des binaires (ici «c:\tmp\VTK\binary»)[18].
- Toujours dans l'invite de commande de Msys, taper «make» afin de lancer l'installation de VTK, l'opération prend plusieurs heures.
- VTK est maintenant installé, il faut ajouter à la variable d'environnement «PATH» le chemin d'accès au répertoire bin de VTK (ici «c:\Program Files\VTK\bin»).

1.4 Création du Projet

Toutes les librairies sont maintenant installées, mais leur utilisation demande certaines manipulations spécifiques. Nous expliquerons la création d'un projet pour Eclipse C++, mais il est possible de reproduire la procédure pour d'autres logiciels de développement. Voici les étapes à suivre :

- Avec Eclipse, créer un projet C++, introduire un nom (par exemple «Segment»), sélectionner le type «Makefile project»->«Empty project» et sélectionner la chaine d'outil «MinGW GCC».
- Sur l'écran suivant, apppuyer sur «Advanced setting».
- Sélectionner «C/C++ Build», décocher «use default build command» et insérer la commande suivante «make -C PATH_TO_PROJECT\Build VERBOSE=1 -j»[19].
- Sélectionner le sous-onglet «Settings» de «C/C++ Build» et cocher «March-O Parser».
- Appuyer sur «OK» et «FINISH».
- Importer les sources dans le projet mais en ne les insérant pas dans un répertoire «Build» (par exemple les mettre dans «src»).
- Créer une MakeTarget sur le projet.
- Créer un fichier «CMakeList.txt» dans le répertoire du projet et le compléter comme dans la section suivante.

[18] Dans ce cas il faut taper la ligne de commande «cd /c/tmp/VTK/binary».
[19] Où PATH_TO_PROJECT est le chemin d'accès au répertoire contenant le projet C++.

- Avec CMake, sélectionner le répertoire des sources du projet et le répertoire des binaires (à mettre dans le fichier du projet, par exemple dans «Build»).

- Configurer et générer de la même façon que précédemment.

- Le projet peut maintenant être compilé.

1.4.1 CMakeList

Voici la structure générale que doit avoir le fichier CMakeList.txt:

```
// indique la version minimale de cmake à utiliser
cmake_minimum_required(VERSION 2.4)
// indique le nom du projet
PROJECT(NOM_DE_PROJET)

SET (CMAKE_VERBOSE_MAKEFILE ON)
SET (BUILD_SHARED_LIBS ON)

FIND_PACKAGE(ITK) //retrouve les librairies ITK
IF(ITK_FOUND)
INCLUDE(${ITK_USE_FILE})
ELSE(ITK_FOUND)
MESSAGE(FATAL_ERROR «ITK not found. Please set ITK_DIR.»)
ENDIF(ITK_FOUND)

FIND_PACKAGE(VTK) //retrouve les librairies VTK
IF(VTK_FOUND)
INCLUDE(${VTK_USE_FILE})
ELSE(VTK_FOUND)
MESSAGE(FATAL_ERROR «VTK not found. Please set VTK_DIR.»)
ENDIF(VTK_FOUND)

FIND_PACKAGE(QT) // retrouve les librairies Qt
IF(QT_FOUND)
INCLUDE(${QT_USE_FILE})
ELSE(QT_FOUND)
MESSAGE(FATAL_ERROR «QT not found. Please set QT_DIR.»)
ENDIF(QT_FOUND)

// inclusion des fichiers ITK, VTK et QT
INCLUDE_DIRECTORIES(${ITK_LIBRARY_DIRS})
LINK_DIRECTORIES(${ITK_LIBRARY_DIRS})
INCLUDE_DIRECTORIES(${VTK_LIBRARY_DIRS})
LINK_DIRECTORIES(${VTK_LIBRARY_DIRS})
INCLUDE_DIRECTORIES(${QT_LIBRARY_DIRS})
LINK_DIRECTORIES(${QT_LIBRARY_DIRS})

// la liste des fichiers sources à compiler
ADD_EXECUTABLE(NOM_DE_PROJET
                    SOURCE1.CPP
                    SOURCE2.CPP
```

```
                ...
                )

// lie les librairies nécessaires
TARGET_LINK_LIBRARIES(NOM_DE_PROJET
        LIB1
        LIB2
        LIB3
        ...
        )
```

2 Installation

Cette partie aborde l'installation du logiciel de segmentation. Les différents cas de figure y sont traités. En effet, le logiciel peut être utilisé seul, mais comme il a été développé dans le cadre médical, il se doit de s'intégrer aux logiciels déjà en place. C'est pour quoi l'installation avec Telemis (logiciel médical) est abordée.

Il est important de mettre les fichiers .dll, le fichier contenant les images et le répertoire contenant l'atlas dans le répertoire du logiciel. Sans les fichiers .dll, le logiciel ne se lancera pas. Si les images sont manquantes, il n'y aura aucune icone dans l'application. Enfin, si le logiciel ne retrouve pas l'atlas, il ne pourra exécuter la segmentation.

2.1 Avec Telemis

2.1.1 Automatique

Lancer le fichier « install.bat ».

2.1.2 Manuelle

Copier le fichier de configuration du plugin « Segmentation.plugin.telemis.cfg » dans le répertoire suivant :

« C:\Program Files\telemis\telemis\entity\tmrhe\cfg\imageProcessingPlugins\ »

À l'emplacement désigné par <thirdPartySoftware> dans le fichier de configuration du plugin, créer un fichier « launch.bat » de manière à indiquer l'emplacement de l'exécutable du programme de segmentation ainsi que celui du répertoire contenant la série d'images cibles et le répertoire de destination :

« start <path_to_exec> <source_directory> <destination_directory> <atlas_directory> »

<source_directory> doit correspondre au répertoire <outputPath> spécifié dans le fichier CFG de configuration du plugin. <destination_directory> doit correspondre au répertoire <inputPath> spécifié dans le fichier CFG de configuration du plugin.

2.2 Sans Telemis

Créer un fichier « launch.bat » de manière à indiquer où se trouve l'exécutable du programme de segmentation et où se trouve le répertoire contenant la série d'images cibles ainsi que le répertoire de destination :

"<path_to_exec>" <source_directory> <destination_directory> <atlas_directory>

3 Description Générale

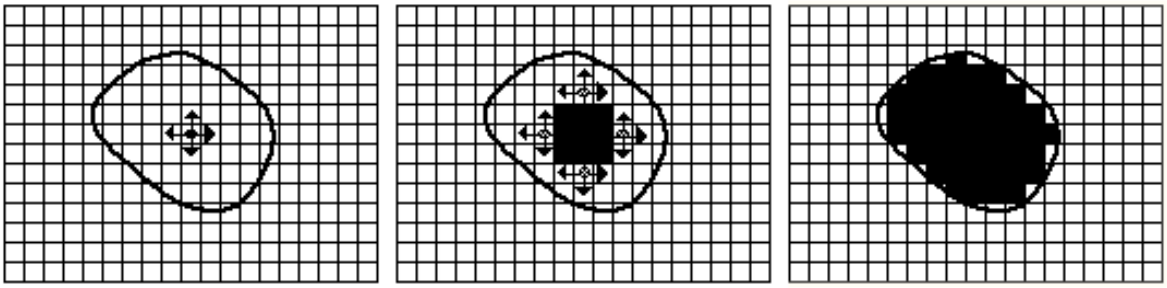
Le logiciel décrit dans ce document est un outil de traitement d'images à caractère médical. Il a été créé afin d'identifier, d'isoler et de visualiser une structure anatomique d'une série d'images au format DICOM, cela dans le but de permettre aux médecins spécialisés dans cette structure anatomique de :

- Détecter les anomalies dans la structure ;
- Localiser les anomalies de la structure ;
- Mesurer l'ampleur des anomalies ;
- Quantifier la structure anatomique étudiée.

La structure anatomique étudiée est le cartilage du genou.

3.1 Principe Général

Pour atteindre les objectifs fixés, l'outil propose de partir d'un (ou plusieurs) point(s) particulier(s) de l'image, appelé(s) marqueur ou graine, qui appartien(nen)t à la structure à isoler. Ce point est le premier paramètre de la méthode. Ensuite, on procède à l'agrégation progressive des pixels voisins du marqueur sur base d'un critère de similitude de manière à former une région de pixels homogène et connexe. Le critère de similitude est exprimé sous la forme d'un intervalle de niveaux de gris qui caractérise au mieux la structure à isoler sur la série d'images. Cet intervalle est le second paramètre de la méthode et doit être établi par l'utilisateur.



Pour établir le(s) point(s) de départ, l'outil possède deux techniques.

La première est automatique et travaille sur les données en trois dimensions. L'outil détermine les marqueurs de départ dans un atlas. Il s'agit du résultat obtenu sur un sujet et qui a été accepté par des experts. L'atlas est déformé afin de correspondre au mieux à la structure étudiée et afin que les marqueurs choisis tombent dans cette même structure.

La seconde est interactive et travaille sur les données en deux dimensions, coupe par coupe. Dans ce cas, l'utilisateur devra sélectionner un ou plusieurs point(s) supplémentaire(s) et relancer la

méthode. Un mécanisme permettant de propager le marqueur de départ de coupe en coupe a été intégré à l'outil. Ce mécanisme repose sur l'hypothèse que le déplacement et la déformation de la structure d'intérêt entre deux coupes consécutives sont minimes et calcule les marqueurs de la coupe n à partir de la surface isolée sur la coupe n-1.

Remarque 1: il est essentiel que le marqueur possède une valeur d'intensité comprise dans l'intervalle de niveaux de gris défini sinon la croissance de la région sera stoppée immédiatement.

Remarque 2: plus l'intervalle de niveaux de gris défini est large, plus la région a tendance à s'étendre au travers de l'image globale et au-delà des limites de l'objet d'intérêt que la région avait pour but d'isoler.

3.2 Correction

Le problème le plus fréquemment rencontré lors du traitement de la série est le problème de fuite. Ce problème se traduit par un débordement de la région d'intérêt au sein d'autres tissus adjacents dû à la présence de discontinuités dans le contour de la structure à isoler. Deux méthodes de correction ont été implémentées.

3.2.1 Automatique

La détection des fuites se fait par la comparaison de la segmentation obtenue par l'outil avec l'atlas. Puisque l'atlas a été déformé pour correspondre au plus avec le sujet étudié, les différences entre l'atlas et le sujet sont donc minimes. Une fuite sera donc détectée par une forte augmentation de la zone segmentée chez le sujet.

La correction s'effectue alors sur base d'une enveloppe à l'extérieur de laquelle, il est peu probable que la zone segmentée soit du cartilage. L'enveloppe est créée pour prendre en compte les légères différences résiduelles entre l'atlas déformé et le sujet étudié.

3.2.2 Manuelle

Des fuites résiduelles peuvent encore être présentes après la correction automatique, c'est pourquoi il existe un outil de correction manuelle.

Il s'agit d'un outil de sélection et de suppression de la zone non désirée. La sélection est interactive, par la pose des sommets d'un polygone englobant la zone non désirée.

Un autre problème peut exister pour certains sujets, des parties de la région d'intérêt n'ont peut-être pas été sélectionnées. Dans ce cas, l'utilisateur devra ajouter un marqueur dans la partie oubliée par le programme.

3.3 Gestion des fichiers

Le logiciel nécessite de nombreux fichiers pour son bon fonctionnement, cette partie a pour but d'expliquer leur utilité ainsi que l'endroit de leur stockage. Commençons par les fichiers requis avant l'exécution :

- Pour commencer, le logiciel a besoin de la série d'images à segmenter. Cette série d'images doit être de format DICOM et se trouver dans le répertoire source décrit plus haut. Soit par défaut :

« C:\fromTelemis\ »

- En plus de la série à segmenter, le logiciel a besoin de son atlas. L'atlas est installé en même temps que l'application mais on peut le modifier. L'atlas est constitué de deux séries d'images DICOM. La première représente l'organe sur lequel l'atlas est basé et doit se trouver dans le répertoire suivant :

« C:\Segmentation\atlas\1\medicale\ »

La seconde représente la segmentation de cet organe et doit se trouver dans le répertoire suivant :

« C:\Segmentation\atlas\1\segmentation\ »

Lors de l'exécution de l'application, plusieurs fichiers sont générés afin d'avoir une trace des différentes étapes de la segmentation. En segmentation 2D, aucun fichier n'est sauvegardé mis à part le résultat final. La description des fichiers suivants concerne donc les étapes de la segmentation en 3D.

- **med-volume.mhd et .raw :** représentent le genou à segmenter en 3D. (Le préfixe "med" signifie que se sont des images médicales et non binaires.) Emplacement : *C:\Segmentation\tmp*.

- **med-atlas-volume.mhd et .raw :** représentent le genou de l'atlas en 3D. Emplacement : *C:\Segmentation\tmp*.

- **bin-atlas-volume.mhd et .raw :** représentent la segmentation du genou de l'atlas en 3D. (Le préfixe "bin" signifie que se sont des images binaires et non médicales.) Emplacement : *C:\Segmentation\tmp*.

- **med-atlas-resampled.mhd et .raw :** représentent le genou de l'atlas en 3D dont on a modifié le pixel spacing pour le faire correspondre à celui du genou à segmenter. Emplacement : *C:\Segmentation\tmp\resampling*.

- **bin-atlas-resampled.mhd et .raw :** représentent la segmentation du genou de l'atlas en 3D dont on a modifié le pixel spacing pour le faire correspondre à celui du genou à segmenter. Emplacement : *C:\Segmentation\tmp\resampling*.

- **med-shrinked.mhd et .raw :** représentent le genou à segmenter en 3D dont on a réduit les dimensions (cette étape est nécessaire à cause de la grande consommation de mémoire de l'algorithme de déformation). Emplacement : *C:\Segmentation\tmp*.

- **med-atlas-shrinked.mhd et .raw :** représentent le genou de l'atlas en 3D dont on a réduit les dimensions. Emplacement : *C:**Segmentation**tmp*\.

- **fields.mhd et .raw :** représentent le champ de vecteurs qui permet de déformer l'atlas vers le genou à segmenter. Emplacement : *C:**Segmentation**tmp**deformation*\.

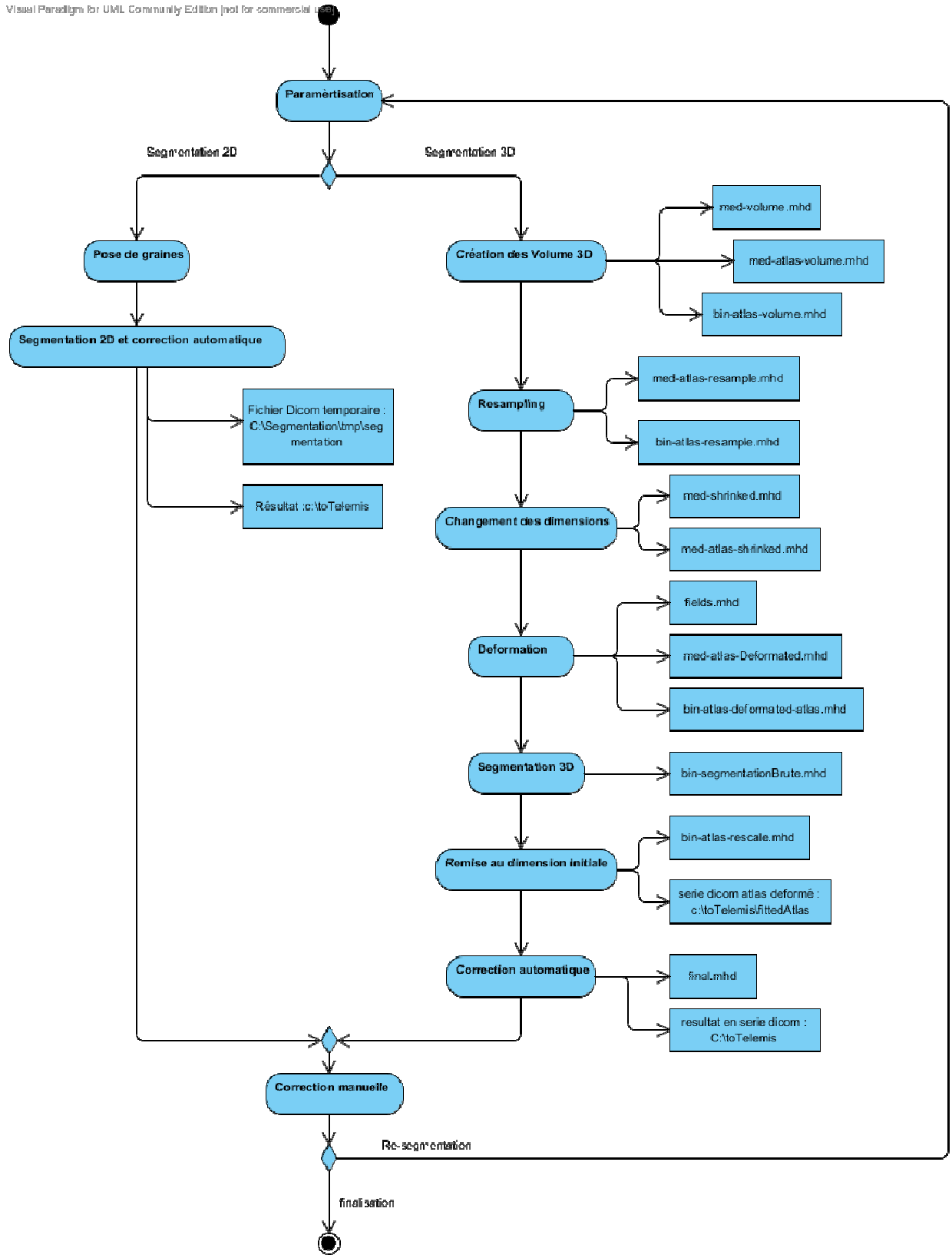
- **med-atlas-Deformated.mhd et .raw :** représentent le genou de l'atlas auquel on a appliqué le champ de vecteurs. Emplacement : *C:**Segmentation**tmp**deformation*\.

- **bin-atlas-deformated-atlas.mhd et .raw :** représentent la segmentation du genou de l'atlas à laquelle on a appliqué le champ de vecteurs. Emplacement : *C:**Segmentation**tmp**deformation*\.

- **bin-atlas-Rescale.mhd et .raw :** représentent la segmentation du genou de l'atlas déformé remis aux dimensions originelles (cette étape est indispensable pour la superposition lors de l'affichage de l'atlas). Emplacement : *C:**Segmentation**tmp**rescaling*\.

- **bin-segmentationBrute.mhd et .raw :** représentent la segmentation du genou sans correction. Emplacement : *C:**Segmentation**tmp**segmentation*\.

- **final.mhd et .raw :** représentent la segmentation du genou avec correction. Emplacement : *C:**toTelemis*\.



Des fichiers « **final.mhd** » et « **bin-atlas-Rescale.mhd** », deux séries DICOM sont créées. La première est le résultat de la segmentation, elle porte le même nom que le genou qui est à segmenter en plus du suffixe « result » et son emplacement est :

« C:\toTelemis\ »

La seconde représente l'atlas déformé pour correspondre au genou à segmenter, elle porte le même nom que le genou qui est à segmenter en plus du suffixe « atlasDeformed » et son emplacement est :

« C:\toTelemis\fittedAtlas\ »

3.3.1 Fichiers binaires

Les fichiers appelés binaires ci-dessus ne le sont pas réellement. Ce sont des fichiers ne comportant que quatre valeurs qui permettent l'affichage de quatre couleurs pour l'overlay. Voici la liste des valeurs ainsi que leur couleur associée et leur signification.

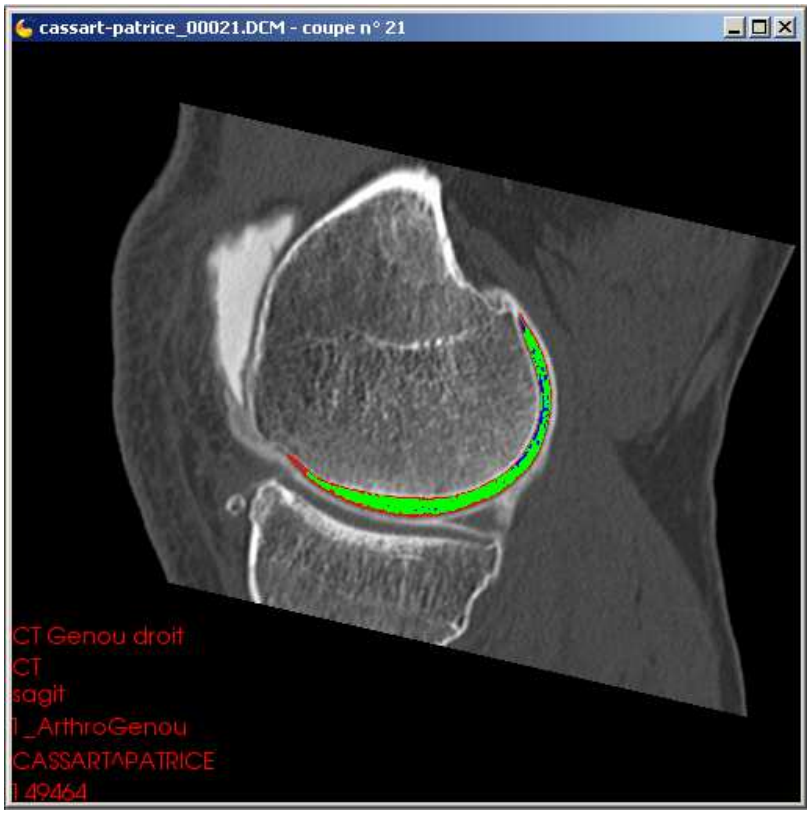
- Valeur **0** : Cette valeur représente l'absence de segmentation et est donc transparente.

- Valeur **255** : Cette valeur représente la segmentation du cartilage dont l'intensité de couleur correspond à l'intervalle du cartilage dans l'échelle de Hounsfield (c.à.d: [50;200]). Cette valeur est représentée par la couleur verte.

- Valeur **240** : Cette valeur représente la segmentation du cartilage dont l'intensité de couleur est supérieure à l'intervalle du cartilage dans l'échelle de Hounsfield. Cette valeur est représentée par la couleur rouge.

- Valeur **230** : Cette valeur représente la segmentation du cartilage dont l'intensité de couleur est inférieure à l'intervalle du cartilage dans l'échelle de Hounsfield. Cette valeur est représentée par la couleur bleu.

  Exemple de représentation :

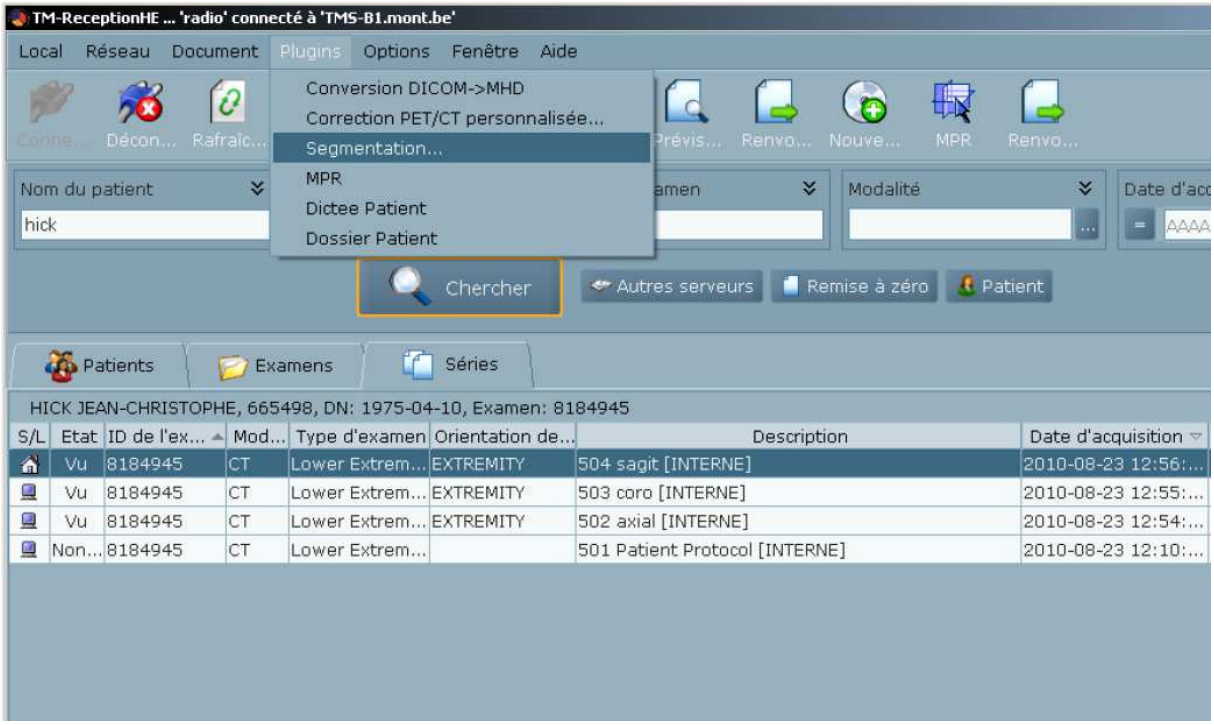


4 Utilisation

Cette section aborde l'utilisation du logiciel de segmentation.

4.1 Lancement avec Telemis

- Lancer Telemis
- Sélectionner le patient cible ;
- Sélectionner l'examen cible ;
- Sélectionner la série d'images cible ;
- Dans le menu "Plugins" de Telemis, sélectionner "Segmentation...".

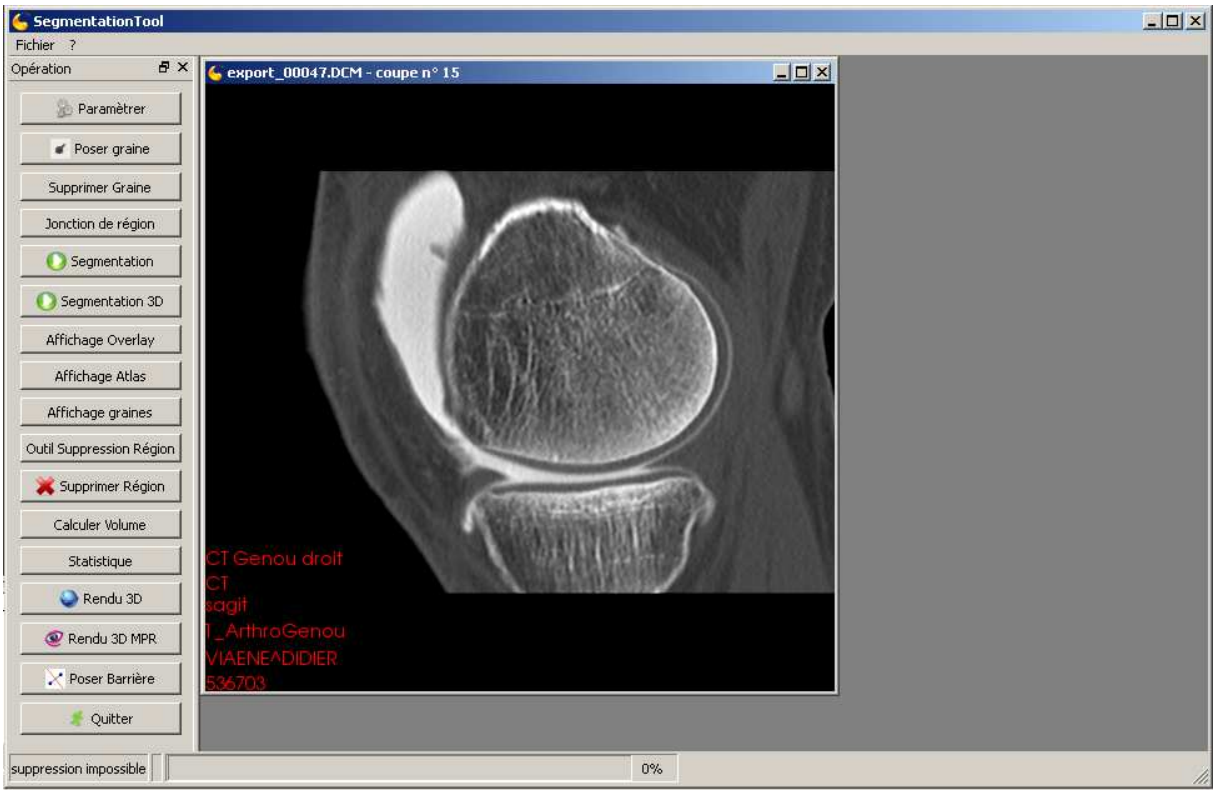


4.2 Lancement sans Telemis

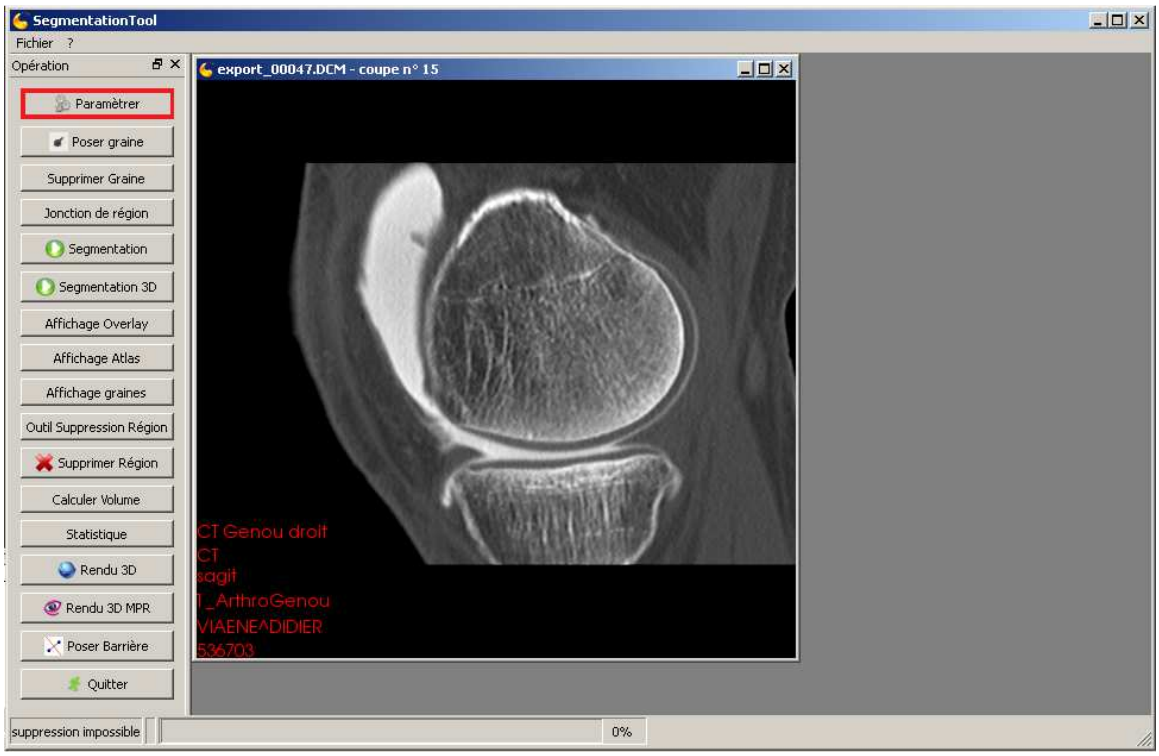
Après avoir placé la série d'images DICOM à traiter dans le répertoire source de l'application, double-cliquer sur "Launch.bat".

4.3 Mode d'emploi

Une fois l'application lancée, la fenêtre principale met quelques secondes avant d'apparaître. Cela est dû au chargement des images DICOM. La fenêtre principale est de la forme suivante :

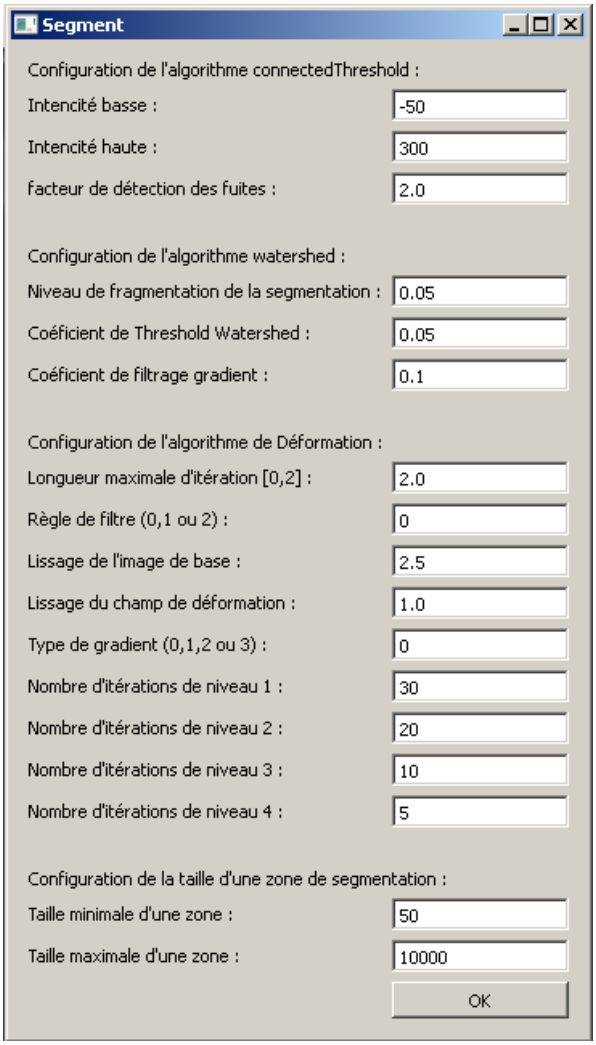


Avant d'aller plus loin, les différents algorithmes utilisés par le logiciel doivent être paramétrés. Pour cela, il suffit de cliquer sur le bouton "Paramètrer" :



4.3.1 Paramétrisation et segmentation

Ce bouton affiche la fenêtre de paramétrisation des algorithmes. La fenêtre contient déjà une configuration de base qui donne de bons résultats dans les cas les plus courants.



Le premier bloc de champs configure l'algorithme de croissance de région utilisé en 2D et 3D. Le second bloc configure l'algorithme de "Watershed" utilisé uniquement en 2D afin de raffiner le résultat. Le troisième bloc configure l'algorithme de déformation de l'atlas et le quatrième bloc configure la taille minimale et maximale des zones segmentées.

- **Intensité basse :** Ce champ permet de configurer la borne inférieure de l'intervalle de niveaux de gris correspondant au cartilage. La valeur minimale est de -1024, elle représente la couleur noire et correspond à l'air. Plus la valeur de ce champ est basse, plus l'algorithme du « region growing » accepte de pixels. Ce champ doit toujours avoir une valeur inférieure à celle du champ **Intensité haute**.

- **Intensité haute :** Ce champ permet de configurer la borne supérieure de l'intervalle de niveaux de gris correspondant au cartilage. La valeur maximale est de 2000, elle représente la couleur blanche et correspond à des matières résistantes telles que les os ou le métal. Plus la valeur de champ est élevée, plus l'algorithme du « région growing » accepte de pixels.

- **Facteur de détection des fuites :** Ce champ représente le coefficient multiplicateur maximal entre la segmentation et le modèle avant d'appliquer une correction automatique. Le modèle est la coupe précédente en segmentation 2D ou la coupe de l'atlas en segmentation 3D. Plus le facteur est élevé, plus une différence importante sera nécessaire pour que la correction ait lieu.

- **Niveau de fragmentation de la segmentation :** Ce champ indique le niveau de fragmentation voulu dans la segmentation finale par l'algorithme du « watershed ».

- **Coefficient de threshold watershed :** Ce coefficient spécifie la différence entre les niveaux de gris pour que des pixels adjacents ne soient pas inclus dans la même zone.

- **Coefficient de filtrage gradient :** Ce champ représente le coefficient du filtre qui est appliqué avant la segmentation par « watershed » afin de lisser l'image.

- **Longueur maximale d'itération :** Ce champ spécifie le temps d'une itération de calcul du champ de déformation. La valeur zéro indique qu'il n'y a aucune limite.

- **Règle de filtre :** Ce champ spécifie le type de règle utilisé pour le calcul du champ de déformation. La valeur 0 utilise le filtre *Diffeomorphic exponentiel*, la valeur 1 utilise le filtre de base d'ITK *fast symetric* et la valeur 2 utilise le *Diffeomorphic Thirion's proposal*.

- **Lissage de l'image de base :** Ce champ spécifie le coefficient de lissage de l'image de base.

- **Lissage du champ de déformation :** Ce champ spécifie le coefficient de lissage du champ de déformation.

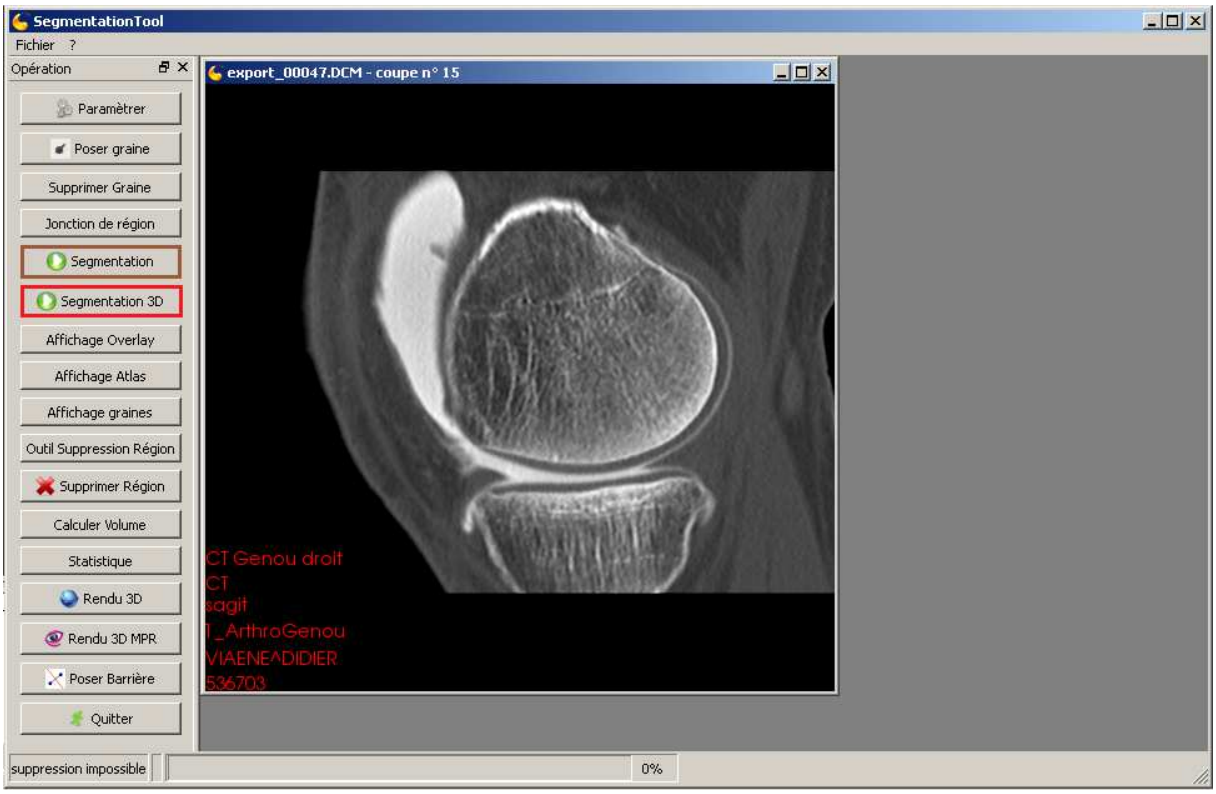
- **Type de gradient :** Ce champ spécifie le type de lissage appliqué aux images. La valeur 0 utilise le « symmetrized (ESM for diffeomorphic and compositive) », la valeur 1 utilise le « fixed image (Thirion's vanilla forces) », la valeur 2 utilise le « warped moving image (Gauss-Newton for diffeomorphic and compositive) » et la valeur 3 utilise le « mapped moving image (Gauss-Newton for additive). »

- **Nombre d'itérations de niveau 1, 2, 3 et 4 :** Ces champs spécifient le nombre d'itérations de calcul du champ de déformation. Chaque niveau représente un niveau de définition. Le niveau 1 est le plus bas et le niveau 4 est la définition la plus élevée. Le dernier niveau travaille donc davantage sur le détail. Le premier niveau doit avoir une valeur strictement supérieure à zéro. Les autres niveaux peuvent avoir une valeur égale à zéro, indiquant dès lors que ce niveau de détail ne sera pas traité.

- **Taille minimale d'une zone :** Ce champ spécifie la taille minimale en pixels qu'une zone doit avoir. En dessous de cette limite, la zone est effacée.

- **Taille maximale d'une zone :** Ce champ spécifie la taille maximale en pixels qu'une zone de segmentation peut avoir. Au-delà de cette taille, la zone est effacée.

L'utilisateur n'est pas obligé de modifier la configuration de base, mais il est nécessaire de confirmer la configuration. Une fois la configuration fixée, l'utilisateur a le choix entre lancer une segmentation dite 3D (cadre rouge) ou une segmentation 2D (cadre brun).

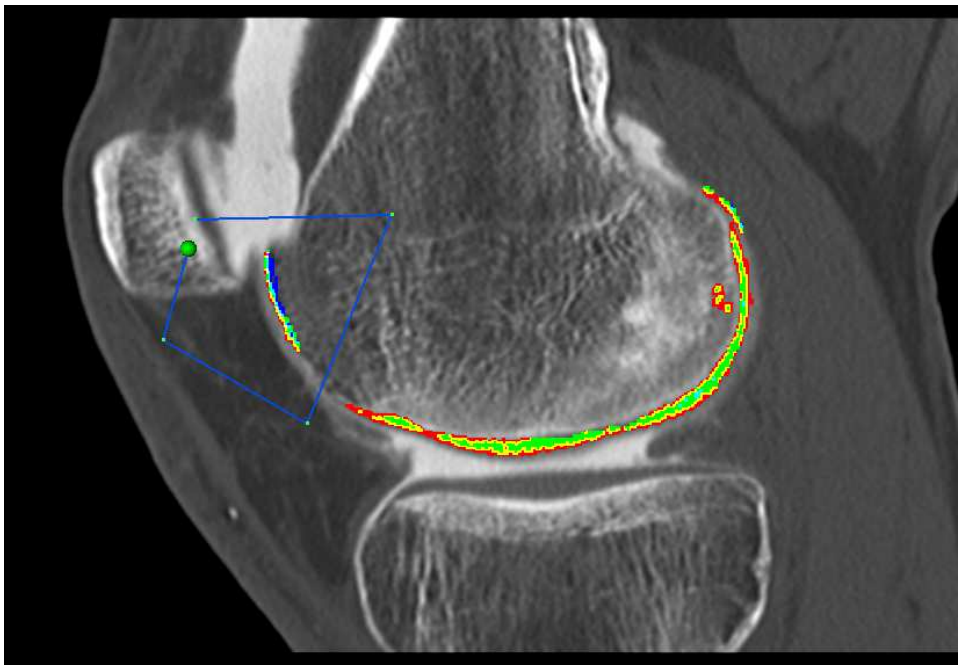


Dans le cas d'une segmentation 2D, l'utilisateur devra introduire une ou plusieurs graines sur les coupes désirées par un clic gauche. Il devra pour cela utiliser le bouton "poser graine". Un simple clic permet de poser une graine sur l'image, un double clic permet de poser un nombre infini de graines sur les coupes désirées. Pour désactiver la pose de graine en double clic, il suffit de recliquer sur le bouton. En mode double clic, le bouton est coloré en bleu.

Une fois la segmentation lancée, l'utilisateur devra en attendre la fin avant de pouvoir interagir avec l'application. Une fois la segmentation terminée, l'utilisateur peut visionner le résultat en parcourant les coupes de la série. Pour cela, il utilisera la molette de la souris. Il pourra agrandir ou réduire l'image à l'aide du clic droit et d'un mouvement en avant ou en arrière de la souris.

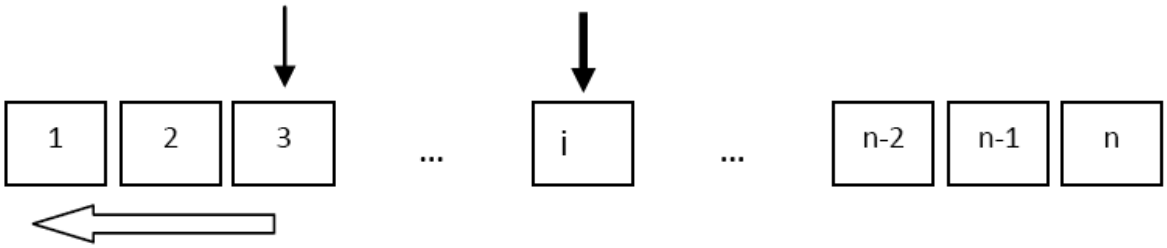
4.3.2 Correction

Si l'utilisateur identifie une zone qui n'aurait pas dû faire partie de la segmentation, un outil de suppression est disponible. Pour supprimer la zone en question, l'utilisateur devra cliquer sur le bouton "Outil Suppression Région" pour activer l'outil. Il devra ensuite déterminer interactivement une région à supprimer et cliquer ensuite sur le bouton "Supprimer Région". La région incluse dans le cadre bleu sera supprimée.

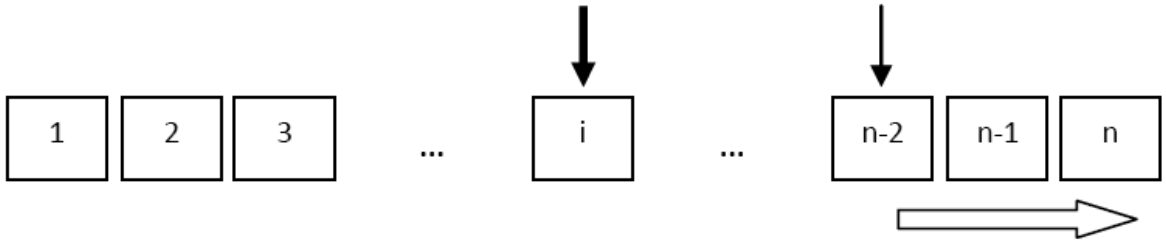


Si l'utilisateur décide de relancer une segmentation, son application sera différente en fonction du type de segmentation lancée. Si une segmentation 3D a été demandée, la correction n'influencera que la coupe sur laquelle elle a été appliquée. Par contre dans le cas d'une segmentation 2D, cela dépend de la coupe de départ de la première segmentation.

Si la correction a été apportée sur une coupe avant la coupe de départ, la correction sera appliquée sur les coupes précédentes.

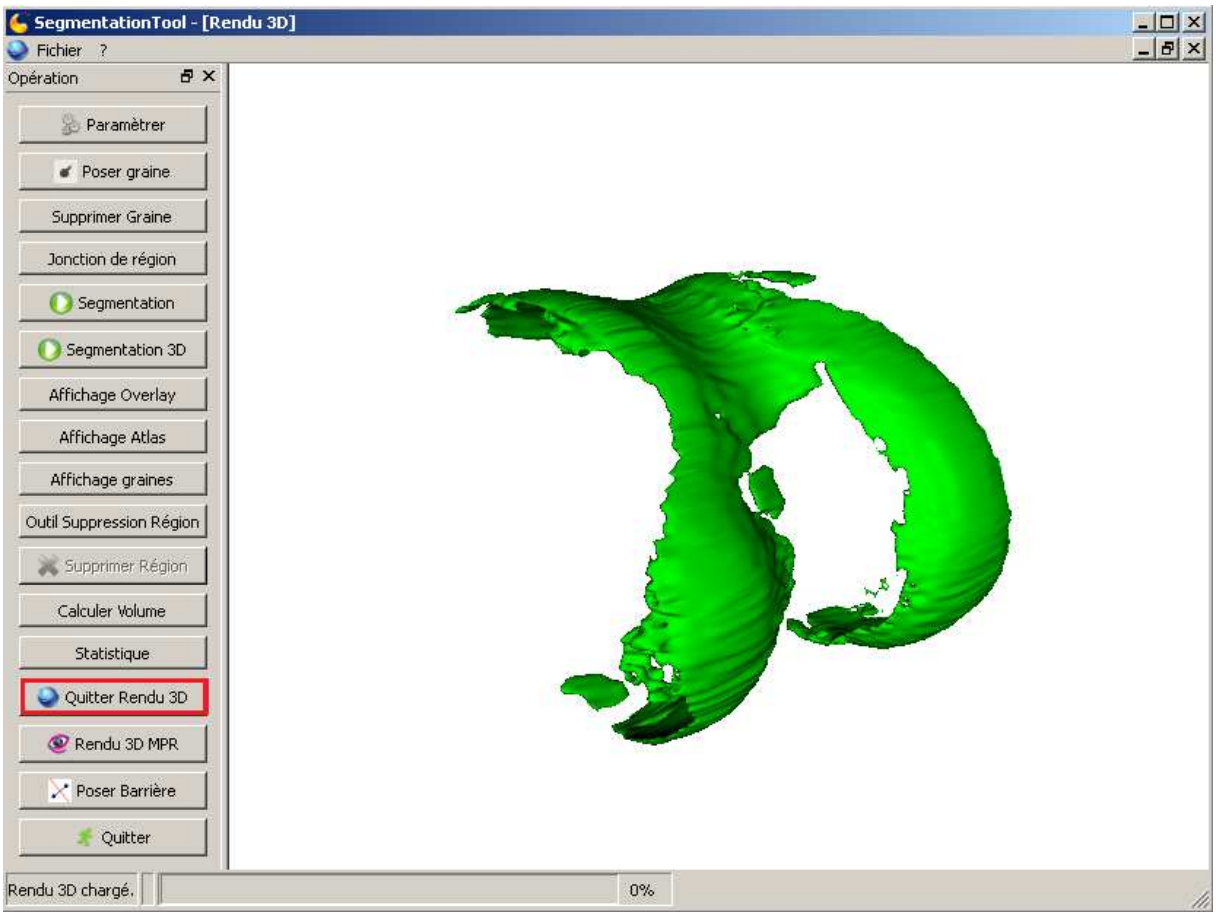


Si la correction a été apportée sur une coupe après la coupe de départ, la correction sera appliquée sur les coupes suivantes.



4.3.3 Rendu

Une fois que la segmentation obtenue est satisfaisante, l'utilisateur peut visualiser le volume de la segmentation. Pour ce faire, il doit cliquer sur le bouton "Rendu 3D". Pour le fermer, il devra cliquer sur "Quitter Rendu 3D" et ne pas utiliser la croix.



De même, pour quitter proprement l'application, l'utilisateur devra utiliser le bouton "Quitter" dans la liste des opérations ou dans le menu déroulant "Fichier".

Bibliographie

[1] K. Moore et A. Dalley, Anatomie Médicale Aspects fondamentaux et applications cliniques \ 2e édition, paris: De Boeck Université, 2007.

[2] «cartilage,» Wikipedia, 5 Juin 2012. [En ligne]. Available: http://fr.wikipedia.org/wiki/Cartilage. [Accès le 12 Juillet 2012].

[3] Y. Juvain, P. Roux, M.-P. Levallois et M. Masson, Petit Larouse de la médecine, Paris: Larousse, 2002.

[4] «L'imagerie médicale,» Wikipedia, 9 Juin 2012. [En ligne]. Available: http://fr.wikipedia.org/wiki/Imagerie_m%C3%A9dicale. [Accès le 11 juillet 2012].

[5] E. Moerschel et J.-P. Dillenseger, Guide des technologies de l'imagerie médicale et de la radiothérapie : Quand la théorie éclaire la pratique, Elsevier Masson, 2009.

[6] S. Wilfart, Médecine Nucléaire : Développement d'un outil de traitement spécialisé pour la segmentation d'image médicale 3D et validation dans le domaine de l'ostéoarticulaire, Namur, 2011.

[7] NEMA, «DICOM homepage,» [En ligne]. Available: http://medical.nema.org/standard.html. [Accès le Avril 2012].

[8] KitWare, «MetaIO Wiki,» [En ligne]. Available: http://www.itk.org/Wiki/MetaIO/Documentation#Quick_Start. [Accès le Avril 2012].

[9] KitWare, «ITK about,» [En ligne]. Available: http://www.itk.org/. [Accès le Avril 2012].

[10] KitWare, «VTK about,» [En ligne]. Available: http://www.vtk.org/. [Accès le Avril 2012].

[11] Nokia, «Qt - A cross-platform application and UI framework - A propos,» [En ligne]. Available: http://qt.nokia.com/about-fr. [Accès le Avril 2012].

[12] «Region growing,» [En ligne]. Available: http://en.wikipedia.org/wiki/Region_growing. [Accès le Avril 2012].

[13] W. K. Pratt, Digital image processing: Piks Scientific Inside4th edition, WILEY-INTERSCIENCE, 2007.

[14] «watershed (image processing),» [En ligne]. Available: http://en.wikipedia.org/wiki/Watershed_%28image_processing%29. [Accès le Avril 2012].

[15] I. Sebari et D.-C. He, «Les approches de segmentation d'image par coopération région-contours,» chez *Revue Télédétection, vol. 7, n° 1-2-3-4*, EDITIONS SCIENTIFIQUES GB, 2007, p.

499–506.

[16] R. Torsten, R. Brandt, R. Menzel, D. B. Russakoff et C. R. J. Maurer, «Quo Vadis, Atlas-Based Segmentation?,» chez *The Handbook of Medical Image Analysis: Segmentation and Registration Models*, Kluwer, 2005.

[17] T. Vercauteren et F. Dru, *An ITK Implementation of the Symmetric Log-Domain Diffeomorphic Demons Algorithm,* 2009.

[18] T. Vercauteren, X. Pennec, A. Perchanta et N. Ayacheb, *Diffeomorphic Demons Using ITK's Finite Difference Solver Hierarchy,* 2007.

[19] J.-P. Thirion, «Image matching as a diffusion process: an analogy with Maxwell's Demons,» chez *Medical Image Analysis volume 2*, Oxford University Press, 1998, p. 243–260.

[20] T. Vercauteren, X. Pennec, A. Perchanta et N. Ayacheb, *Non-parametric Diffeomorphic Image Registration with the Demons Algorithm.*

[21] KitWare, «paraview about,» [En ligne]. Available: http://www.paraview.org/. [Accès le Avril 2012].

[22] Telemis, «produit pour hôpitaux et cliniques,» [En ligne]. Available: http://www.telemis.com/for_healthcare_institutions.html. [Accès le Juillet 2012].

[23] KitWare, «CMake about,» [En ligne]. Available: http://www.cmake.org/. [Accès le Avril 2012].

[24] KitWare, «MinGW about,» [En ligne]. Available: http://www.mingw.org/. [Accès le Avril 2012].

[25] Eclipse, «wascane Project home,» [En ligne]. Available: http://code.google.com/a/eclipselabs.org/p/wascana/. [Accès le Avril 2012].

[26] S. Calande, Médecine Nucléaire : Détection de tumeurs dans la région pulmonaire sur base d'acquisitions PET/CT synchronisées, Namur, 2009.

[27] T. Vercauteren, X. Pennec, A. Perchanta et N. Ayacheb, *Diffeomorphic Demons: Efficient Non-parametric Image Registration,* Neuroimage, 2008.

[28] J. Sachs, «http://ftp2.bmtmicro.com/dlc/Resampling.pdf,» 2001. [En ligne]. Available: http://ftp2.bmtmicro.com/dlc/Resampling.pdf.